Jeremy Odell

Principal Architect | AI Systems Engineer | Technical Leader

jeremyodell@gmail.com • (713) 306-3956 • Fulshear, Texas

"Building systems that matter—where technical excellence meets human impact."

Professional Identity

I'm a hands-on architect who writes code daily—not the kind who only draws diagrams. With 25 years of progressive experience from Computer Programmer through Principal Architect, I've built enterprise systems where failure has real consequences: energy infrastructure serving millions, retail operations affecting families' budgets, financial fraud detection protecting consumers, and AI platforms transforming developer productivity.

My career has been defined by a few core patterns: I build production systems at scale, I mentor teams while staying technically deep, and I bridge the gap between complex technology and business value. Whether architecting event-driven systems processing millions of messages per hour, building RAG systems from scratch, or leading cross-functional initiatives, I focus relentlessly on outcomes that matter.

Core Technical Expertise

Enterprise Architecture: 25 years building distributed systems, microservices architectures, event-driven platforms, and multi-tenant SaaS applications. Deep expertise in TOGAF frameworks, ArchiMate modeling, and strategic technical roadmaps.

AI & Machine Learning Systems: Production experience building LLMs and intelligent agents from scratch—not just API integration. Orchestrated sub-agent workflows, RAG systems with vector databases, custom MCP servers, and AI governance frameworks. AWS Certified AI Practitioner.

Java & Spring Ecosystem: 25 years Java development from J2EE through modern Spring Boot microservices. Deep Spring Framework expertise including Spring AI, Spring Boot, Spring MVC, and RESTful API design patterns.

Cloud & Infrastructure: AWS certified with extensive experience in Bedrock, SageMaker, CloudFormation, Lambda, containerization (Docker/Kubernetes), and Infrastructure as Code. Multi-cloud experience including GCP.

Event-Driven Architecture: Built Enterprise Service Bus processing millions of messages/hour. Expert in ActiveMQ, Kafka, RabbitMQ, asynchronous messaging patterns, and distributed system design.

Full-Stack Development: Backend (Java, Python, Node.js), Frontend (Angular, React, Vue.js), Data (PostgreSQL, MongoDB, Snowflake, pgvector), and everything in between. Led 300-page legacy Swing→Angular SPA modernization.

How I Learn and Grow

My Learning Philosophy

After 25 years in software engineering, I've learned that the most valuable skill is learning itself. Technology changes constantly—Java frameworks evolve, new AI paradigms emerge, cloud platforms release new services—but the ability to rapidly acquire deep expertise remains constant. My approach to learning combines intellectual curiosity with pragmatic hands-on practice: I don't just read about new technologies, I build with them.

I'm driven by what I call 'productive obsession'—when I encounter a technology that could solve real problems, I dive deep until I truly understand it. This happened with Spring AI when I needed to build RAG systems at Freepoint. It happened with orchestrated agent workflows when I saw how AI could transform development at APG&E.; It happened with event-driven architectures when I needed to process millions of messages at Energy Transfer. Each time, I went beyond surface-level understanding to build production systems that delivered measurable value.

My Learning Process

When approaching new technology, I follow a deliberate learning process:

1. Understand the 'Why' First: Before diving into implementation details, I understand the problem the technology solves and why it exists. For RAG systems, I studied why retrieval-augmented generation improves LLM accuracy before building the vector database architecture. This context helps me make better architectural decisions.

2. Build From First Principles: I don't just use frameworks—I understand what they do under the hood. When building orchestrated agent workflows, I didn't just use LangChain patterns blindly. I architected custom solutions using Spring AI, understanding state management, error handling, and agent communication protocols from scratch. This depth means I can debug production issues and make informed trade-offs.

3. Learn by Teaching: The best way to solidify learning is teaching others. At APG&E;, I didn't just deploy Claude Code—I mentored 9 developers on AI-assisted development patterns, established best practices, and documented approaches for organizational adoption. Teaching forces me to organize knowledge clearly and identify gaps in my understanding.

4. Stay Current Through Doing: I maintain technical currency by continuously building. Whether it's experimenting with new AWS services, trying different AI frameworks, or exploring emerging patterns in distributed systems, I learn by implementing production-quality code—not just reading documentation.

Recent Learning Highlights

Generative AI & LLM Systems (2021-Present): When ChatGPT emerged, I recognized this wasn't just hype—it was a fundamental shift. I dove deep: earned AWS AI Practitioner certification, built production RAG systems from scratch using Spring AI and PostgreSQL pgvector, architected orchestrated multi-agent workflows, and developed custom MCP servers. I didn't just learn to call APIs—I learned to architect intelligent systems that reason, retrieve context, and accomplish complex goals.

Cloud-Native Architecture Evolution: Continuously evolving my cloud expertise beyond basic 'lift and shift.' Mastered Infrastructure as Code with CloudFormation, implemented containerized microservices with Kubernetes, and architected serverless event-driven systems. Each project deepened my understanding of cloud-native patterns.

Modern Frontend Frameworks: While my foundation is backend, I led the modernization of a 300-page Java Swing application to responsive Angular SPA. This required learning modern TypeScript, component-based architecture, reactive programming patterns, and Material Design—demonstrating I can quickly master new domains when needed.

How I Approach Domain Knowledge Gaps

Throughout my career, I've successfully entered new domains where I lacked specific knowledge. I don't have healthcare experience, but I've built mission-critical systems in energy, finance, and technology where the same principles apply: reliability, accuracy, security, and user experience. I don't have direct retail experience, but I architected platforms at Freepoint supporting retail energy operations by rapidly learning business workflows and translating them into scalable technical solutions.

When entering a new domain, I follow a systematic approach: immerse myself in the domain by speaking with subject matter experts, understand the business processes and pain points deeply, identify how technology can create value, and build solutions that feel natural to domain experts—not generic tech imposed on their workflows. This approach has worked whether the domain is energy trading, fraud detection, or AI-enabled development.

Commitment to Continuous Growth

At this stage of my career, I'm not content to rely on past expertise. The past 4 years diving deep into AI/ML while maintaining my enterprise architecture foundation demonstrates my commitment to staying at the technology forefront. I read technical papers, experiment with emerging frameworks, contribute to technical communities, and maintain hands-on coding skills.

What excites me about the next phase of my career is applying 25 years of proven expertise to new challenges while continuing to learn emerging technologies. Whether it's advancing in AI systems architecture, exploring new cloud paradigms, or entering new industry domains, I approach growth with intellectual curiosity tempered by pragmatic judgment. I know what I know deeply, I recognize what I need to learn, and I have proven processes for rapidly acquiring expertise that delivers value.

Leadership Through Technical Excellence

I lead through demonstrated expertise, not title. Whether mentoring 9 developers at APG&E;, leading a team of 5 engineers at Freepoint, or introducing new methodologies at Energy Transfer, I earn credibility by showing teams how it's done—writing production code, debugging complex issues, and making tough architectural decisions alongside them.

My mentorship approach focuses on sustainable growth: I set clear technical standards, demonstrate by example, provide constructive code reviews, create learning opportunities, and celebrate improvement. At Cognizant, this approach reduced post-release defects by 50%. At APG&E;, it accelerated development cycles

by 25%. These weren't mandates—they were cultural transformations achieved by making teams better.

I believe in kind, not nice: providing tough feedback clearly and constructively, challenging assumptions respectfully, holding high standards while supporting growth, and celebrating learning from failures. This creates environments where technical excellence flourishes.

Core Professional Values

Hands-On Excellence: I write production code daily. Architecture without implementation is theory. I stay technically deep because it makes me a better architect, mentor, and leader.

Measurable Impact: I focus relentlessly on outcomes. 25% faster development cycles, 40% query performance improvement, 50% defect reduction, zero-data-loss migrations—I measure success by value delivered, not features shipped.

Intellectual Honesty: I acknowledge what I don't know, I change my mind when presented with better information, and I'm transparent about trade-offs. This builds trust and enables better decisions.

Continuous Learning: After 25 years, I'm still learning daily. The moment I stop being curious is the moment I stop being effective.

Team Success Over Individual Recognition: My greatest achievements are the teams I've built and the developers I've mentored. Making others better is how I multiply my impact.

Mission-Driven Work: I want my software to matter—whether helping care teams save lives, enabling families to manage electricity needs, or empowering developers to build better systems faster.

This is who I am: a hands-on architect with deep technical expertise and proven leadership who continues learning, growing, and building systems that create real value. I bring 25 years of experience solving complex problems, but I approach each new challenge with the curiosity and energy of someone still excited to learn. Whether architecting AI-native applications, mentoring development teams, or diving into new domains, I'm ready to contribute technical excellence combined with continuous growth.