RED | LINE

A REDLINE ADVISORS FIELD REPORT · NO. 02

Coding at the Speed of *Thought.*

What a 30-Year Software Veteran Learned in 45 Days.

Mick Hollison

FOUNDER & CEO, REDLINE ADVISORS

2026 · REDLINECS.AI

THE OPENING

I lost fifteen hours without *noticing.*

It started at 7 AM in my home office in Park City, Utah. When I next looked up, it was approaching 10 PM.

I had barely eaten or even stood up. My Apple Watch had been buzzing at me to stand for hours. That is not usually a problem for me. It was almost as if I wasn't even in the room.

What I had been doing was building software.

Not specifying software. Not reviewing mockups. Not sitting across a table from an engineer, trying to describe what was in my head. Building it myself. Page by page. Component by component. Typography detail by typography detail. I was working for a client called CodeVine, who is about to launch the world's first *Agentic Flow Platform* into a market that doesn't know it needs one yet.

When I finally pushed back from the keyboard, I did not feel tired. I felt the way I used to feel in my twenties, walking out of a competitive sales bake-off I had just won. Exhilarated. And, more importantly, a little stunned.

"Coding was someone else's job."

Here is the thing about me you should know. I am not a developer. I have not been a developer for thirty years. I spent the intervening decades as a software marketer, a sales leader, a product executive, a Chief Marketing Officer, and eventually the President of a public software company you've probably heard of. Its name is Cloudera.

I graduated Georgia Tech in 1990 with a management degree, a Co-op year at GTRI (Georgia Tech Research Institute), and more time at the keyboard than the major suggested. At GTRI I had digitized employee HR records on a Compaq lunchbox with an amber screen, dual 5¼-inch floppies, and a copy of dBase. Later in college I helped a small Atlanta business that printed logo'd merchandise for clients like HBO build their seasonal catalog on a Mac with Aldus PageMaker, work that had previously cost them a

fortune at a design firm. Ahead of me was a job at a small Atlanta startup called Samna. For the next five years I wrote just enough Ami Pro macros to build custom features for some of the biggest law firms in Washington. Then, like most people who end up running software companies, I stopped writing code altogether.

Forty-five days ago, I sat down with Claude, Anthropic's coding agent. I did not realize, at the time, that I was beginning six of the most productive weeks of my career.

I am writing this on day forty-five. I started twelve hours ago. I have not stopped.

This is what happened.

01 · THE NUMBERS

What forty-five days *looked like.*

Before you hear the story, see the scoreboard.

Here is what I built. A full corporate website for my own firm, RedLine Advisors, from scratch. Every page. Every component. Every animation. A marketing and product site for CodeVine, the client mentioned above, at the care-level my teams at Cloudera applied to billion-dollar product launches. A full internal CRM, called Signal Engine, that replaced what would otherwise have been years of HubSpot subscriptions and invoices. A competitive battlecard application for a B2B AI platform client, now templated so I can stand the same app up, rebranded, for any future client in an afternoon. A custom audit report template for a well-known software testing firm, also productized for reuse. A branded rate-card PDF builder. A prospect-PDF generator. A methodology for building client brand books from scratch in a day. A field report. The article you are reading now.

45 DAYS Calendar window	213 COMMITS Three codebases	25,747 LINES OF CODE Net, shipped, standing	21.6 ACTIVE HOURS Hands-on keyboard
1.44B TOKENS Context exchanged	84K WORDS BY ME A small book	1.5M WORDS BY CLAUDE Roughly fifteen novels	6 PRODUCTION APPS Shipped & templated

213 git commits across three production codebases. **25,747 net lines of code** that I wrote, reviewed, shipped, and in many cases rewrote, with Claude as my partner. *Net* means added and still standing. Not just typed.

1.44 billion tokens of context exchanged between me and Claude. If you prefer the pre-AI unit, that is roughly the equivalent of a small public library's worth of text moving back and forth between my laptop and Anthropic's servers, on my couch, at Utah altitude.

21.6 hours of actual hands-on keyboard time with Claude over those forty-five days. Not *"had Claude open in a tab."* Not *"checked email while a build ran."* Twenty-one hours of live, continuous back-and-forth between a human and an agent whose only subject was the thing I was making.

I wrote roughly **84,000 words** of prompts, specifications, objections, and revisions to Claude, across Claude Code and Claude.ai combined. That is a small book's worth of instruction. Claude wrote back roughly **1.5 million words** of code, prose, and analysis. That is the rough equivalent of fifteen novels. The output-to-input ratio, measured in words, was better than **fifteen to one**. This is not because I am a gifted writer of prompts. It is because Claude is an astonishingly capable writer of everything else.

Those are the numbers for me. Running a consulting firm, not coding full-time. Client meetings and flights and the ordinary chaos of work life continuing around me the whole time.

Those numbers do not reflect how well I code.

I am not a programmer. I am an executive who has spent his career learning how to describe problems and solutions in clear, cogent language. To customers, boards, and investors. To employees, analysts, and the press. Really, to just about anyone who could help me achieve my goals.

That skill (well-structured, effective communications) is the single most transferable one into this new way of working. Claude is an astonishingly capable writer of software. My job, the throughput above would suggest, is to describe what the software is supposed to do with astonishing clarity.

The numbers reflect vivid descriptions, not simply keystrokes. That distinction is everything.

What coding *used to be.*

For most of the thirty years I worked in software, coding was a black art.

It was a discipline practiced by a small class of people who were often brilliant, usually idiosyncratic, and occasionally heroic. They sat at a different table, wore different t-shirts, kept different hours, and spoke in a language the rest of us had agreed to pretend we mostly understood. I ran marketing for them. I ran sales for them. I eventually helped run an entire public software company built on what they made. I sat in product reviews and design reviews and launch rehearsals and quarterly operating meetings where we all nodded respectfully as the engineering leader held up a Jira dashboard and told us which features would slip, by how much, and why.

The rest of us had something we wanted the software to do. They had the power to decide whether, when, and how that would happen.

In that world, software was something you bought. You lobbied a product team to build it. You put it on a roadmap. You waited a year for a small army of engineers to ship it, pixel by pixel, sprint by sprint. Everything useful had already been made by somebody else, and it was *rented* at a per-seat-per-month price in perpetuity. When you wanted the product to do something it didn't do, you filed a ticket. Or you bought a second product to bridge the gap. Or you hired a consultant.

I knew this world so well I stopped noticing it, the way fish, presumably, stop noticing water.

Every savvy instinct I had told me that the one thing I was never going to do, whatever else I picked up along the way, was write the software myself. Coding was for Navy SEALs. I was, by then, an aging admiral.

Five teachers, one unexpected *curriculum.*

They all thought they were teaching me how to run software companies. None of us knew they were training me to make software myself.

01	02	03 / CITRIX	04 / CTX→CLDR	05 / CLDR
Bill Jones SAMNA	Charlie Pappas LOTUS	Mark Templeton PRES → CEO	Dave Moxey PARTNER	Paul Codding EVP PRODUCT
"Don't let the dying category define the emerging one." DREAM HIGHER	*"Custom is achievable. It's a language. Languages can be learned."* THE POSTURE	*"Every slide. Every icon. Every pixel. Every detail."* ATOMIC DETAIL	*"Shipping is a discipline, not a moment. Challenge every process."* RIGOR	*The product we didn't ship was also, it turns out, training.* UNFINISHED

Bill Jones · *Samna*

Bill was the lead product manager at the Atlanta startup I joined straight out of Georgia Tech. He knew everything about Samna's flagship product. It was called Ami Pro: a Windows word processor that, radically for the time, let you see formatting on screen exactly as it would print. It shipped before Microsoft Word for Windows existed. It was visibly, flagrantly, several years ahead of its category. Lotus acquired us shortly after I joined because the technology was the best in the market at the time.

Bill taught me something beyond the product itself. He taught me that you do not let the standards of the dying category define what's possible in the emerging one. WordPerfect was the giant at the time. Bill refused to let WordPerfect's feature list become the ceiling for what a word processor could be. He dreamed higher. He expected the rest of us to dream higher too, or to get out of the way.

Charlie Pappas · *Lotus*

Charlie was a product specialist and my technology partner at Lotus. He was a University of Georgia grad. I had gone to UGA's century-old in-state rival, Georgia Tech. One of us was contractually required to carry water for the other side. As much as I hate to admit this, it was usually me. Charlie was a very gifted developer.

Charlie taught me two things, in order. First, the Ami Pro macro language. Then, a few years later, he taught me Lotus Notes. It was a collaborative application platform that anyone who worked in enterprise software in the 1990s will recognize. At the time, it was the most sophisticated platform of its kind on the market.

Big law firms in Washington would ask for things our products didn't quite do. Top-tier shops like Patton Boggs and Shaw Pittman. Charlie and I would go off and write custom features for them, first in Ami Pro macros and later in Notes. It was not exactly computer science. But for a few years, I wrote a little code. Nothing major. Nothing magical. But real hands-on-keyboard work nonetheless. I watched code do things for customers that changed how they worked. Then I became a marketer, then a manager, then a director, then an executive. Coding became, as I said, someone else's job.

(Charlie, for what it's worth, eventually walked away from software entirely. He is now a successful photographer in Florida. Life has many good shapes.)

What I did not understand at the time, and what I am only now understanding, is that Charlie wasn't really teaching me Ami Pro or Notes. He was teaching me that the distance between *"I wish this software did X"* and *"I just made this software do X"* is not a mystical gap. It's a language. Languages can be learned. And even when the language leaves your active memory, what stays behind is the posture. The assumption that custom is achievable.

Mark Templeton · *Citrix*

Mark probably shaped me more than any other single person in my career. He is a CEO who thinks like a product manager, and a product manager who thinks like a designer, and a designer who thinks like an obsessive. Every pixel. Every icon. Every error message. Every word on every slide of every keynote. We shipped XenDesktop together. Shortly after my team built it, it became the undisputed champion of its category — desktop virtualization, or VDI — and materially changed Citrix's growth trajectory. In the run-ups to our flagship customer event, Synergy, Mark and I would work until two or three in the morning and be back at it by six. We sweated the details, at *an almost atomic level*.

I did not know it then, but that is exactly the skill coding in the AI-era requires.

Not typing speed. Not algorithm knowledge. Not CS theory. The ability to break a problem down to its atomic unit and describe that unit with absolute precision. Then verify that what was produced matches what was asked for. As it turns out, decades of sweating keynote slides was actually great training for writing software. It was training for sweating components, API contracts, type definitions, state transitions, edge cases, error paths, and every pixel of every button on every page.

Dave Moxey · *Citrix → Cloudera → RedLine*

Dave is my longtime partner in crime across three companies. In the spirit of full disclosure, I'm also honored to call him one of my closest friends. He was a product manager when I met him. Over the years I helped pull him more firmly into product marketing, where today he is a textbook product-and-marketing operator. The trade was fair. He taught me rigor. I taught him positioning. I fly by the seat of my pants. Dave ships methodically. He taught me to challenge received processes. He taught me that shipping is a discipline, not a moment. Most of the rigor I have is Dave's rigor. Twenty years of co-pilot seat time with him is how I managed, in forty-five days of solo flying with Claude, not just to have a good time — but to actually ship.

Dave is largely retired these days, though he still drops into RedLine from time to time to help us land the big swings. The rest of the time, he is applying his disciplined rigor to maintaining a low single-digit handicap on the golf course. The trade, again, seems fair.

Paul Codding · *Cloudera*

Paul led our Product Management team at Cloudera, one of the early leaders in the Big Data and Analytics space. He didn't work directly for me on the org chart, but he led our "Horizon 3" product work: the experimental, innovative bets that might never reach the market. He was also the most talented PM I have ever worked with, a genuine savant. Paul and I built a product together at Cloudera that, for reasons that no longer matter, never shipped. I remain convinced it would have been a genuine breakthrough. Today, I think about it every time something *does* ship. Especially something that would have taken our team at Cloudera six months and a dozen engineers.

Paul is now leading the product organization at Sema4.ai, well positioned at the bleeding edge of the agentic AI movement.

The product Paul and I didn't ship was also, it turns out, great training.

The moment *it shifted.*

Back to the fifteen hours.

On the day in question, I was building a comprehensive, content-rich website for a brand new company called CodeVine. It needed every detail Mark Templeton used to demand. Typography, spacing, iconography, animation timing, interaction copy, mobile breakpoints, accessibility, motion preferences. The works. For twenty years, I had worked with design firms who would quote one-hundred thousand dollars to build the brand and another one-hundred and fifty-thousand dollars to build the site.

I built the site in a week, with Claude. And oh yeah, it didn't cost the client a fortune to do it either.

What happened on the fifteen-hour day was nothing short of miraculous. My old posture had a specific voice. It said, *"I know what I want; I'll go explain it to the engineers."* That voice was suddenly useless. I was the one doing the explaining now. Claude was the one doing the building. The two of us were running the loop together, in real time, with nobody in between.

I wanted a hero section with a particular kind of growing-vine animation, in a forest green, with paths drawn in a specific sequence at a specific easing curve, with nodes that popped in at keyframe timestamps I could describe to the hundredth of a second. I described it. Claude wrote it. It was wrong in three small ways. I described the three small ways. Claude fixed them. We looked at it together. I asked for it slower. It was slower. I asked for the green to be warmer. It was warmer. I asked for the nodes to appear only after the branch they sat on had finished drawing. They did.

In any prior era of my career, this conversation would have been three Zoom calls, two revised mockups, one disappointed design firm, and a week of calendar time. It took approximately four minutes.

I looked up fifteen hours later.

What I had been experiencing without knowing it was the dissolution of every filter I had ever worked through. In the old world, my instinct had to travel through a long chain of humans. Through a product manager, through a designer, through an engineer, through a reviewer, through a QA tester, through a deployment. It arrived, often weeks later, partially intact. Each filter was a human. Each human had a backlog. Each human had an opinion. Each filter took a little of the signal with them.

With Claude, there were no filters. There was my intent, described precisely, and there was the product on my screen, now.

Not just faster. Fundamentally different.

The skills were not new. Templeton taught me the skills at Citrix. Bill Jones taught me the skills at Samna. Charlie Pappas taught me the micro-syntax in 1992. I had spent thirty years sharpening the ability to say exactly what I meant, and to sweat every detail of whether what got made matched what I had said.

The only thing that had changed was the one thing that matters. The three or four layers of humans between me and the outcome were gone. The filter I had been refining since Samna — the one that could describe what I wanted, precisely and completely, in language a team could execute on — was suddenly connected directly to the hands that would build it.

05 · THE FACTORY

What one person can *build now.*

The numbers only matter if you know what they became.

Redlinecs.ai, the full marketing site for my firm. It includes hero pages, engine pages, solutions pages, Field Reports, a blog, a contact form, and a client vault protected by email-plus-one-time-password authentication. It also has a Claude-powered chat widget that answers questions about our methodology, using our own content as the context it draws from. One hundred thirty-eight commits. Roughly fourteen thousand lines of handcrafted front-end and back-end code. Built from scratch, by me, in the middle of client work.

CodeVine.ai, the full marketing site for a client launching a new category of software. Animated vine illustrations. Platform pages. An ROI calculator. A maturity-curve graphic. A case-studies scaffold. Every page pixel-sweated to the Templeton standard. Built alongside the client team, with their copy and positioning worked through in parallel on Claude.ai, which is a second surface of the same tool, better suited for long-form writing and strategy than for shipping code.

Signal Engine, a full internal CRM for RedLine, replacing a Google Sheet that had become, frankly, embarrassing. Next.js 16 on the front. Supabase on the back. Seventy-five companies, seventy-one contacts, seventy-seven deals. Pipeline view. Deal tables. Live from a real database at signal.redlinecs.ai. Thirty-nine commits. A long weekend of work, plus tuning.

A **Battlecard App** for a B2B AI platform client. A branded, client-editable web application that compares their product to five to seven named competitors, packaged in the visual language of the client's brand system. Built in a hundred-odd Claude turns. Now templated, which is the part that actually matters. I can stand the same app up, rebranded, for any future client in an afternoon. The second one will cost a fraction of the first. The fifth will cost less than the client's lunch.

A **Signal Watch audit report template**. This one captures RedLine's proprietary methodology for measuring a company's messaging clarity against its competitors, rendered as a branded, long-form PDF deliverable. Built first for a well-known software testing firm. Now productized.

A **rate-card PDF builder**. A **prospect-PDF generator**. A **brand-book methodology** I can now execute for any client in a day, based on the one I built for myself. An SOW assignment system. An expense-report engine. Internal tooling. Small, unglamorous, the kind of tooling every consulting firm I've ever met currently pays a subscription fee for. A subscription to a vendor whose per-seat pricing model was designed for a world in which *"seats"* meant people.

Each individual thing is, in the scheme of enterprise software, modest.

The point is not any individual thing. The point is the shape of the inventory.

Here is what happened while I was building. At some point around the battlecard app, I realized I was not just building things. I was building a factory for building things. Each deliverable, done right once, became a template. Each template became an asset. Each asset became a reusable unit of service that I could reapply across every future client at a cost approaching zero.

I did not set out to become a one-person agency. But if you spend forty-five days building websites, internal tools, branded reports, PDF generators, sales enablement apps, and a CRM from scratch, and you are one person, that is what you are. And if every one of those things is now a template, that is what you stay.

I was building a factory for building things.

The hero engineer is the *problem.*

There is a version of this story that does not work out.

In that version, the person who sat down with Claude forty-five days ago was not a seasoned executive who had spent a career thinking about products. It was a working engineer, a good one, maybe even a great one, who looked at Claude, recognized it immediately, and then, crucially, refused to let it change how they worked.

I know this person. You know this person. They are the ten-times-productive individual contributor who has decided, consciously or not, that their productivity is a property, not a system. They figure out a better way to do the thing, and they keep it on their laptop. They work alone. They ship beautifully. They are irreplaceable.

They are also, and this is the part they will not admit, hoarding.

Because the breakthrough each of these engineers is privately having, in their personal editor, on their personal branch, in their personal flow, is exactly the breakthrough I had in my fifteen-hour afternoon. It is the single most valuable artifact their organization has produced that week. And it is sitting on a MacBook Pro that will, statistically, leave the building inside twenty-two months, and go with them.

The 10x engineer has long been treated as a kind of corporate hero. The word everybody uses is *rockstar*.

I think that framing is now the problem.

In the movie *Talladega Nights*, Ricky Bobby and his racing partner Cal call out *"shake and bake"* every time they execute a coordinated move at the front of the pack. The gag only works because both drivers know the call is coming. A move one of them makes without signaling would get them both wrecked. The engineer who insists on being the rockstar, who insists that the leverage they have found belongs to them and not to the org, is running the shake-and-bake play without ever telling the partner. They are doing to their company exactly what our industry used to do to customers. They are hoarding capability in pursuit of leverage. That posture is no longer defensible: the tools to democratize the capability now exist. Individual exceptionalism that refuses to become organizational genius is a ceiling. It is not the point.

The org that wins the next decade isn't the one with the best rockstar engineers. It's the one that figured out how to capture what the rockstars do, and install it into the workflow of everyone else.

I wrote this field report at the same time my friend **Wells Burke** was publishing his own piece, *The Organizational Layer*. Wells is a fellow Georgia Tech alum and one of the founders of CodeVine. His piece is about what it takes for enterprises to capture and compound what their best developers are doing. Mine is about what happens when an individual, coming the other way, discovers the same leverage and ships from his kitchen.

The two pieces are, I think, a single argument seen from two sides. You can read his on CodeVine's site.

07 · THE RESHAPING

The end of *"you have to buy it."*

I built my own CRM in a long weekend.

Let me say that again, because it is the single most disorienting sentence I have written this year.

I built my own CRM in a long weekend.

Signal Engine, the internal pipeline and relationship management tool that runs RedLine Advisors, did not exist on a Thursday evening in early April. By the following Tuesday, it was live at signal.redlinecs.ai, running on Next.js 16 and Supabase, pulling real data from my real deals, and replacing a Google Sheet that had become an embarrassment.

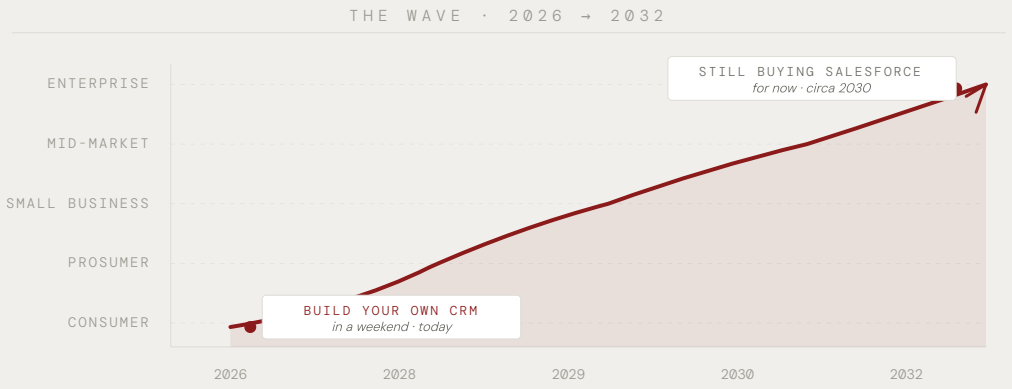
In 2024, the market would have sold me the equivalent of Signal Engine as a HubSpot Sales Hub Professional subscription. About one hundred and fifty dollars per seat per month. Add the implementation consultant I would have needed to wire it up to the rest of my business. Add the annual *"we are raising prices effective immediately"* email every software company sends now. Over three years, for a five-person firm, that adds up to somewhere between thirty and sixty thousand dollars. Plus the loss of all control over how the data is shaped. Plus the loss of all ability to change what the product does. Plus the quiet acceptance that, once again, you will learn to want whatever the vendor has decided to give you.

I do not have to want what the vendor decided anymore.

I built a CRM that does exactly what I need, nothing I don't, and that I can modify in an hour when I need to modify it. It cost me a weekend and what amounts to a rounding error on my Anthropic subscription. I own it outright. Nobody can raise the price on me. Nobody can end-of-life it. Nobody can force me into a migration. I do not have a vendor.

This is not a new thing for large enterprises. Large enterprises have always had custom software. It is called the IT department, and it is rarely efficient. What is new is that *one person* can now have it. A single operator can now live at the custom-software standard that used to be reserved for the Fortune 500.

And this is where we need to talk about pricing. The per-seat SaaS model that built the incumbent software industry was designed for a world in which each seat was a human doing roughly human-scale work. That assumption is already breaking. When agents do the work, the *"seat"* a vendor is charging for becomes a fiction. Most incumbents are not evil. They are simply running the pricing model that got them here. But the model is about to get replaced. The two models that will replace it over the next decade are clearer every month. **Consumption pricing**, where you pay for the tokens, compute, or usage you actually draw down. And **outcomes-based pricing**, where you pay for the result rather than the right to log in. Every major software company in the world is going to have to choose, and the ones that choose slowly are going to feel the ground move.



The buy-what-the-vendor-sells-you era is not over. But the bottom is coming out of it.

FROM THE BOTTOM UP · FOREVER

What comes next is a reshaping of the middle of the software industry. I want to be careful about which parts.

At the top of the market, the big enterprise systems of record (CRMs, ERPs, HRIS platforms, IT service management suites, data warehouses) are not going anywhere fast. The integration depth, the data gravity, the regulatory surface area, and the sheer number of already-entangled workflows make them nearly immovable. Large enterprises will use Claude, and tools like it, to build custom apps on top of those systems and agentic workflows around them. But they will not be ripping out Salesforce in 2027.

At the bottom of the market, every bet is off.

Below the enterprise — at the level of consumers, prosumers, solo operators, small businesses, lean startups, and boutique firms — the ceiling has effectively been removed. A small business like mine can now operate at the scale of a company ten, twenty, perhaps a hundred times its size. We are watching this play out in real time. **Gamma.app** is a small team quietly building an enormous business. Their headcount is a fraction of what such a business would have required five years ago. It is not an edge case. It is the leading edge of a much bigger wave.

The wave will hit consumers first. Then prosumers. Then small businesses. Then mid-market. Then, finally and carefully, the enterprise. It will move in that order, over the next five to seven years, sector by sector, category by category. It will start with the software categories that have always felt like rent you resented paying.

Project management will get reinvented before accounting does. **Monday.com** was new and exciting once. Now it is just another subscription line. It will face a wave of lightweight tools built in a week by the people who were going to pay for Monday. Internal dashboards, basic reporting tools, lightweight approval workflows, simple inventory trackers. These are the categories in which the question is no longer *"which vendor do I buy from?"* but *"do I build or do I buy?"*. And for the first time, *build* is the honest answer.

The buy-what-the-vendor-sells-you era of software is not over. But the bottom is coming out of it. **From the bottom up. Forever.**

08 · THE CALL

Dive right into the *deep end.*

Coding was a black art. A discipline for Navy SEALs, not aging admirals. That wasn't a personal failing. It was just how the world worked.

Software was what you bought, what you lobbied a product team to build, what you waited a year for a small army of engineers to ship.

That world is gone.

For the first time, a vision in my head (the exact product, the exact design, the exact words) can become working software by the end of an afternoon. Not with a small army over months. With a small team aligned on vision, purpose, and the story they want to tell. Sometimes just a team of one.

The friction is gone.

It is the scene in *Limitless*, the one where the pill hits and the brain opens up. Only now it isn't a pill. It is a prompt. It is Trinity in *The Matrix*, learning to fly a helicopter in thirty seconds flat. The capability was always somewhere in your head. You just never had a way to get it out.

You don't have to be twenty-five to see this. You don't have to have been a developer. You don't have to love computers the way engineers love computers. You just have to have the courage to dive in, and to stay in long enough to feel the friction disappear.

Old dogs can absolutely learn this trick. I'm one of them.

Now it's your turn.

ABOUT THE AUTHOR

Mick *Hollison*

Mick Hollison has spent more than thirty years at the intersection of technology, go-to-market strategy, and the messy human art of getting people to believe in something. His career arc reads like a field guide to enterprise tech. He led a worldwide software sales team at **IBM**. He ran global marketing and strategy at **Citrix**. As CMO of **InsideSales.com**, he helped pioneer the application of predictive analytics and big data to B2B sales. And as **President of Cloudera**, he helped guide the company's transformation from open-source startup to enterprise data cloud leader.

In 2023, Mick founded **RedLine Advisors** in Park City, Utah. RedLine helps B2B technology companies defy the algorithm by engineering narratives grounded in the one thing AI will never replicate: *genuine human truth*.

This field report was written on day forty-five of Mick's first full immersion in Claude Code. He started that morning at 7 AM, and is still writing.

RED | LINE

REDLINE ADVISORS

B2B narrative engineering for the AI era. Dive right into the deep end. We will meet you there.

PARK CITY, UTAH

CONTACT@REDLINECS.AI · 435-565-4502

REDLINECS.AI