AI Didn’t Read the Logs

Log Misery Meets AI, the Right Way

Give an engineer a couple of gigabytes of logs per day, mix multiple clients into the same service, then ask for fast answers during a live issue, and people will call that “analysis.”

That's generous.

A better description is selective suffering.

The engineer in this case wasn't looking at neat, well-shaped data. He was dealing with huge volumes of text, day after day, and trying to isolate one specific client’s activity inside a shared service, then narrow further into the right process, the right message type, and the right fields inside those messages. This wasn't broad search. It was surgical log hunting under pressure. In his words, it was deep, cherry-picked work, and it was simply not human-friendly to do quickly in real time.

That's where a lot of conversations about AI go wrong.

The naive move would have been to feed mountains of logs into an LLM and hope it somehow made sense of the chaos. That's the sort of idea that sounds modern right up until it fails, or breaks the bank. What actually worked here was smarter. AI didn't become the analyst of record. It became the accelerator that helped an engineer get far more out of Kibana.

The story isn't that AI replaced expertise. The story is that AI shortened the path to useful expertise.

The engineer didn't start as a Kibana specialist. He started with almost no practical background in Kibana and no prior experience doing this kind of log parsing work. He began with a little introductory training, then started asking AI a better class of question: I have this tool, I have these logs, I need to find this kind of information, correlate it this way, and get to the operational truth faster. From there, AI helped him discover how to configure the tool, how to think about parsing, and how to combine capabilities he would not have found quickly through documentation alone.

One of AI’s most useful roles is helping people learn complex tools faster.

Most learning inside enterprise teams still follows an old pattern. You take some training. You experiment. You make mistakes. You back up. You correct. You make different mistakes. Eventually, after enough friction, you get competent. There's nothing wrong with that model except it’s time-consuming. Most engineers don’t have the luxury of extra hours to explore every capability in every tool they touch. They’re already busy keeping systems running and problems solved.

AI shortened that climb.

The engineer described it well. Instead of spending months learning mainly from failure, he could start from something that worked, then go back and understand why it worked. That's a very different experience. It is more motivating, faster, and far better aligned with how real people learn under deadline pressure. He called it a leapfrog, which is exactly right.

Another lesson sits inside this: teams often own more capability than they use.

Not because they're careless. Not because the tools are bad. But because modern platforms evolve constantly, and working engineers rarely have time to read every release note, study every advanced feature, and experiment with every possible configuration while also keeping systems running. As he put it, people may be doing something the hard way without realizing the tool already offers an easier or more sophisticated path. AI closed that gap.

At first, he handled log parsing himself as part of the learning process, before moving on to the more formal parsing pipeline feeding Kibana. With AI’s help, he moved past basic usage and began combining Kibana capabilities in ways the team hadn't previously been exploiting. The result wasn't just “better dashboards.” It was a different operational model. He built what he called a real-time scanner for individual transactions.

Instead of manually digging through enormous log volumes to reconstruct what happened, the operator can click a button when a process starts and watch the relevant transaction unfold in real time. The scanner shows the stages the process is moving through, the messages being generated, the errors or inconsistencies it finds, whether the process succeeds or fails, where the failure occurs, and which errors led to that failure. It also translates obscure error codes into language people can actually read and understand. That means the result is useful not only to deep technical experts, but to the wider group of people who need to understand what happened and what to do next.

This wasn’t merely a faster search screen. It took away much of the grunt work and let the operator apply their mind to the part of the job that actually matters.

Instead of forcing someone to manually hunt through “dozens of millions of lines” across multiple logs, the system narrows the problem to the transaction that matters and tells its story in motion. It reduces friction and mental load. It makes troubleshooting faster, but just as important, it makes troubleshooting clearer. In operations, clarity under pressure is often worth as much as speed.

The customer noticed. So did the people watching the budget, because this delivered the benefit of AI without the cost shock of round-the-clock analysis on raw logs.

This wasn't polite applause for a clever internal trick. It was a request to use the dashboard across other projects. That's the point where a one-off improvement becomes something more serious. It means the value was visible enough, and practical enough, that others immediately wanted the pattern repeated.

There's a lot of noise right now about AI and productivity. Much of it is shallow. Much of it assumes the point is to hand more work directly to the model. That's often the wrong instinct. In this case, the winning move wasn't asking AI to absorb giant logs and produce magical answers. The winning move was using AI to help an engineer master an existing operational tool quickly enough to change the work itself.

Used well, AI can compress discovery time. It can expose underused capability in tools teams already own. It can reduce the time between “I know what I need” and “I know how to make the system show it to me.” It can help turn a miserable manual task into a repeatable operational advantage.

That’s not hype, it’s leverage.

It’s a far better story than “AI read the logs.” It didn’t.

It threw him a lifeline before he drowned in them.