TECH @ KLOUDMATE / ENGINEERING

We rebuilt the KloudMate AI Assistant v2 to be 4x faster, 75% cheaper and now finds 2.7x more root causes

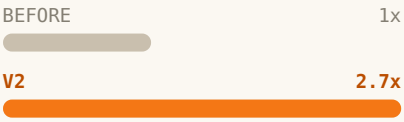
A short version of how we measured it, and what it means for your on-call bill.

By Amitava Saha (CTO) · October 2026

2.7x

more root causes found

Same incidents as the previous version, same recordings, three runs each.



4x

faster chat answers

Matched scenarios. First words appear in two to three seconds.



75%

lower price

Per full investigation. A chat answer is $0.05, down from $0.20.



82%

root cause named, investigation benchmark

<2%

confidently wrong across the full benchmark

19,565

benchmark runs behind these numbers

All values come from KloudMate's internal benchmark. Bars compare the previous and current version on the same recorded incidents. Method notes on the last page.

THE QUESTION WE STARTED WITH

How do we know the next version is better?

The KloudMate AI Assistant answers questions about your systems and investigates your alerts. Version 2 is a ground-up rebuild. We kept one rule through all of it: every change gets measured against real incidents.

An AI assistant is easy to demo and hard to measure. A good answer on Monday says very little about Tuesday's incident. So before we touched the assistant, the team built a benchmark that only moves when the assistant actually gets better.

WHAT CHANGED, AT A GLANCE

2.7x
more root causes found than earlier, on the same incidents

4x
faster chat. 98 seconds down to 24

82%
root cause named on the investigation benchmark

<2%
confidently wrong across the full benchmark

88%
of expected facts met on live failures

75%
lower price. Chat $0.05, investigation $0.25

All values come from KloudMate's internal benchmark. Method notes on the last page.

01 / THE BENCHMARK

The benchmark comes first

We record each incident at one instant, so every version of the assistant sees exactly the same telemetry. The incidents come from three places: our own production systems, faults we injected into a Kubernetes test cluster, and failures we injected into the OpenTelemetry Astronomy Shop, a live online shop with more than 15 services. We also run live tests to keep the recordings honest.

WHERE THE SCENARIOS COME FROM

Our own production
Real incidents from KloudMate systems.

Kubernetes test cluster
Faults we injected on purpose.

OpenTelemetry Astronomy Shop
A live online shop with 15+ services.

Every scenario lists the facts a good answer has to state, like the failed component, the start time and a fix. It also lists what the answer must not claim, like a decoy cause or an old event presented as current. An independent model grades each answer, and we read the graded answers ourselves at every milestone.

HOW EACH CHANGE IS TESTED

1 **Record the incident at one instant** → 2 **Replay it for every version** → 3 **Grade against the fact list** → 4 **Compare with 95% intervals**

Each change runs side by side with the previous version, two or three times per scenario. It only ships if the 95% confidence interval shows nothing got worse. So far that's 19,565 runs, and 13,153 of them have been graded by the judge.

19,565
benchmark runs to date

13,153
graded by the independent judge

95%
confidence interval gate before shipping

02 / THE RULES

Four rules behind every investigation

1 First, accuracy. We set the targets before we started: at least 80% of expected facts as the floor, and 90% as the goal. A change ships only if accuracy holds or improves.

80%
floor: expected facts

90%
goal

2 Second, evidence before conclusions. Like a good SRE, the assistant forms hypotheses and tests each one against traces, logs, metrics and Kubernetes state. Then a separate review checks the conclusion. That review can lower the confidence, but never raise it. If the evidence isn't there, the report says inconclusive.

Hypothesis → **Test on traces, logs, metrics, K8s** → **Separate review** → **Lower confidence or inconclusive**

3 Third, general tools for every stack. The assistant can query any field in your telemetry, the way an engineer would, so it isn't tied to Kubernetes. In our tool-selection test it picks the right tool and fields about nine times out of ten.

9 in 10
times it picks the right tool and fields in the tool-selection test

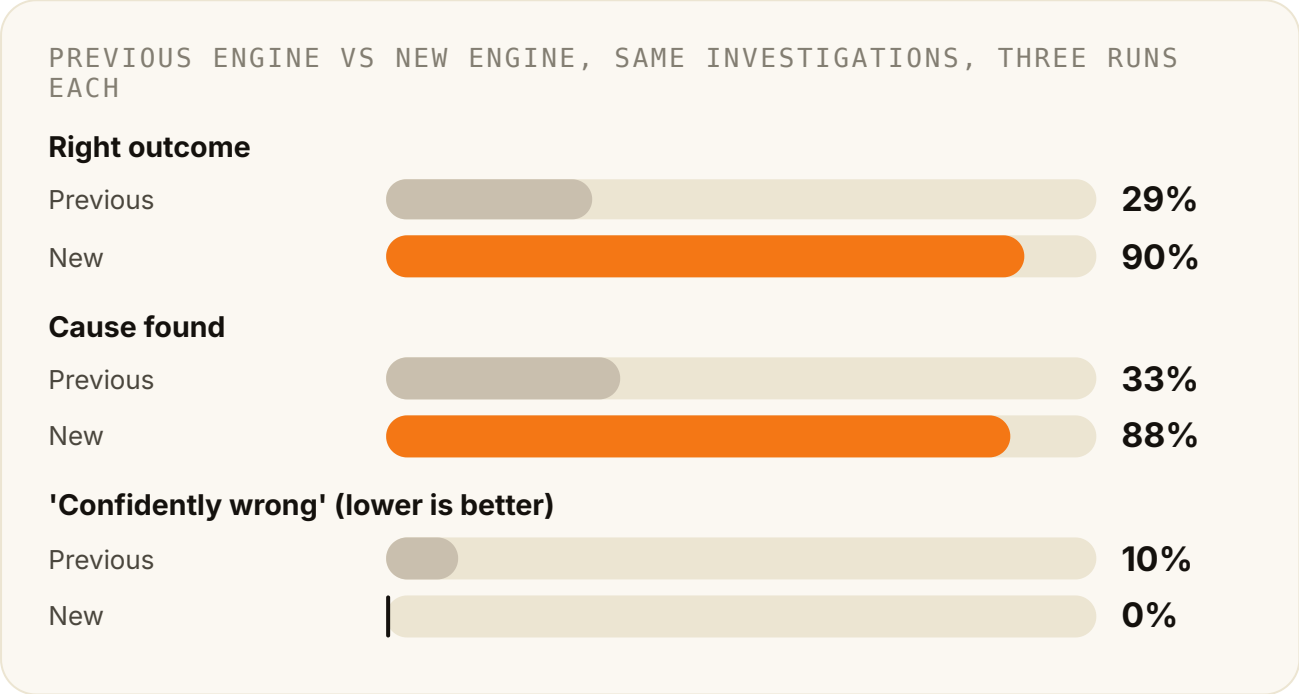
4 Fourth, reports that show their work. Each one has the causal path, a timeline, the causes it ruled out, and the evidence and query behind every claim. Every fix comes with a step to check that it worked.

- Causal path
- Timeline
- Causes ruled out
- Evidence and query per claim
- Step to confirm each fix

03 / CALIBRATION

'Confidently wrong' is the number we watch

An assistant that's sometimes unsure is still useful. One that's confidently wrong sends you down the wrong path. So we track that on its own: how often it names a cause with confidence when the real cause is something else.



10% → 0

confidently wrong answers, matched runs

<2%

confidently wrong across the full benchmark

We ran the new investigation engine against the old one on the same investigations, three runs each, both on our production model. Right outcomes went from 29% to 90%. The share of runs that found the cause went from 33% to 88%. And confidently wrong answers went from 10% to zero: the new engine named a cause in every run, and none of them was confidently wrong. Across the full benchmark, fewer than 2% of investigations are confidently wrong.

That zero comes from one matched comparison. It doesn't mean the assistant can never be confidently wrong.

04 / LIVE FAILURES

Live failures that cascade

A single fault with a clear alert is the easy case. Real incidents cascade. A database stops, and the alert fires from a service three hops away at the edge. So we injected failures into the live Astronomy Shop: services scaled to zero, crash loops, memory limits, bad feature flags, DNS faults, and a database with no free connections. Each investigation started from a different place - the failing component, a downstream service, the edge, or an engineer's question with no alert at all.

FOUR PLACES AN INVESTIGATION CAN START

1. The failing component
2. A downstream service
3. The application edge
4. An engineer's question, no alert

Scaled to zero · Crash loops · Memory limits · Bad feature flags · DNS faults · No free DB connections

88%

of expected facts met

92%

of runs named the exact root cause

More than nine runs in ten.

FAILS THE BAR

"There's a problem with the cart."

PASSES THE BAR

"The cart's cache has no pods because its deployment was scaled to zero."

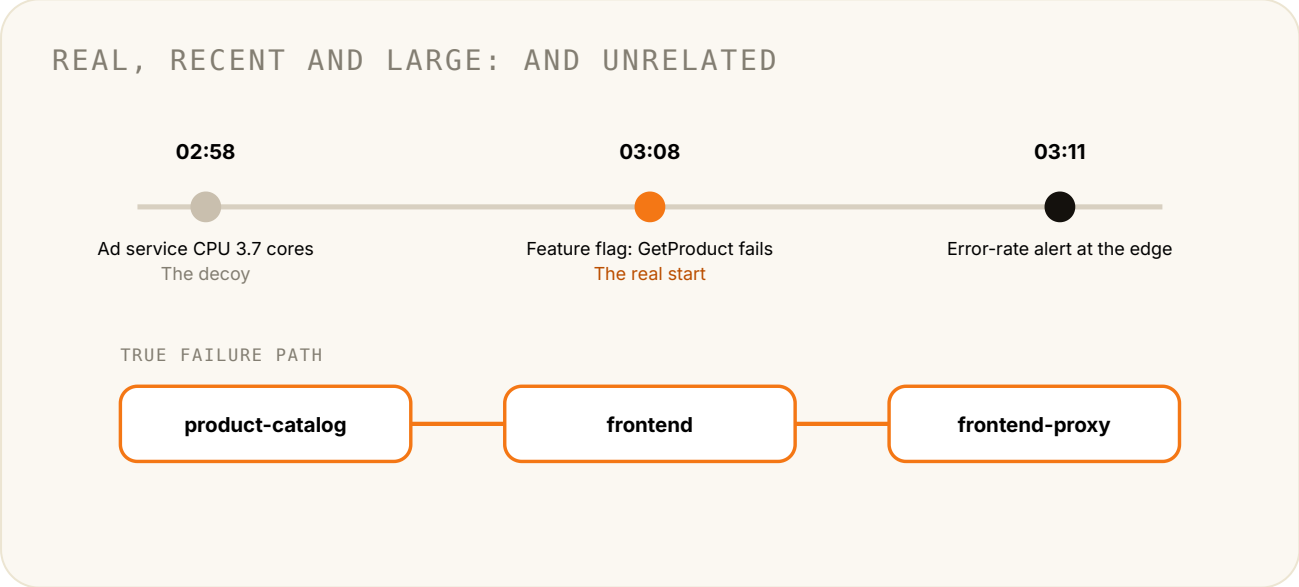
The 82% is a different measurement. It's the share of runs that named the root cause on the wider investigation benchmark, not the exact root cause on the live failures. Different tests, different numbers.

05 / CASE STUDY

The decoy test

Our favourite test plants a red herring. We pushed the ad service's CPU to 3.7 cores. Ten minutes later, a feature flag made some product lookups in product-catalog fail, and the error-rate alert fired at the edge. An assistant that chases the biggest recent anomaly blames the ad service.

The KloudMate AI Assistant followed the failed requests from the edge back to the source. It found the flag, put the start at 03:08, and ignored the ad service. Ten facts out of ten.



10 / 10

expected facts stated

0

blame placed on the ad service

Both runs named the flag, traced the path to the edge and put the start at 03:08.

06 / COST

Why it costs so little

Most of an investigation is fast telemetry queries, not model calls. On average it runs 291 queries and 15 model calls, and takes about four minutes. The model plans the queries and reads the results. Model calls are the expensive part of any AI answer, so using only a handful of them is what makes the new price possible.

ONE INVESTIGATION

291 telemetry queries

15 model calls

About four minutes

Average over benchmark runs

ONE CHAT ANSWER

39 telemetry queries

6 model calls

About 25 seconds

First words in two to three seconds

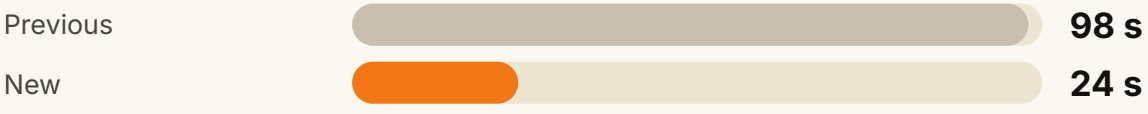
WHERE THE WORK GOES, INVESTIGATION



291 fast telemetry queries — 15 model calls

A chat answer averages 39 queries and six model calls. It takes about 25 seconds, and the first words show up in two to three seconds. Chat answers meet 85% of the expected facts. The four-times-faster number is a separate comparison with the old version, on scenarios both ran: 98 seconds down to 24.

CHAT RESPONSE TIME, MATCHED SCENARIOS, PREVIOUS VS NEW



4x faster

07 / PRICING

The new price

An AI credit now costs $0.05, down from $0.20. A chat answer still uses one credit and an investigation still uses 5. So nothing changes in how you use it. Each credit just costs a quarter of what it did. We're also raising the included monthly credits fourfold. A $10 credit pack now buys 200 credits instead of 50.

PRICE PER CREDIT AND PER ANSWER

PER AI CREDIT

~~$0.20~~

$0.05

PER INVESTIGATION (5 CREDITS)

~~$1.00~~

$0.25

Chat answer: **$0.05** instead of $0.20. Included monthly credits go up **fourfold**. A **$10** credit pack now buys **200** credits instead of 50.

At $1 for an investigation, you save root cause analysis for the big incidents. At 25 cents, you can run it on every alert.

A team asking 20 questions and running 2 investigations every working day

~~$132~~

$33

a month

Automatically investigating 50 alert groups a day

~~$1,500~~

$375

a month

08 / HONEST LIMITS

Where we're still improving

We'd rather tell you the gaps ourselves. Slow resource exhaustion is the hardest pattern we test. A database that gradually uses up its connections looks like ordinary slowness until new connections get refused. Managed cloud services are another one. If the failure is inside a managed service, we have no internal traces to follow, so those score lower.

And it's our own benchmark. We test on systems we know well, and a model does the grading. We review the grades ourselves and keep adding scenarios from real incidents. Next, we're going after larger incidents, where one fault ripples through many services before anyone notices.

HARDEST PATTERN

Slow resource exhaustion, like a database using up its connections.

LOWER SCORES

Failures inside managed services with no internal traces.

Try it on your next alert.

research.kloudmate.com

HOW WE MEASURED

All values come from KloudMate's internal benchmark. "Facts met" is the share of expected facts an answer states correctly, including the claims it must not make. 'Confidently wrong' is the share of investigations whose leading cause has high confidence while the real cause is different; we measured it on the full benchmark at the current version with a second model, which scores below our production model on the same benchmark. The 2.7 times and 4 times comparisons use scenarios both versions ran, from the same recordings, three runs each. Live results come from faults injected into the OpenTelemetry Astronomy Shop. Speed and model-call figures are averages over benchmark runs.