SKG JS MEETUP #9

Your App Is Up.
Your Business Is Down.



Eleftheria Batsou

Developer Advocate, UptimeRobot



Your monitoring is green.

@BatsouElef @eleftheriabatsou



Your monitoring is green.

And your customers can’t pay you.

@BatsouElef @eleftheriabatsou



UptimeRobot

External uptime monitoring for websites, APIs and services.

3.4M+ people watching their services from the outside

Before this: a developer. Both sides of the 2am alert.

@BatsouElef @eleftheriabatsou

THE ONE THING TO LEAVE WITH



The code you wrote

- Watched by every check you have



What your code depends on

- Where most outages actually happen

Most monitoring watches the first. It misses the second.

@BatsouElef @eleftheriabatsou

PART ONE

The twenty minutes nobody noticed



THURSDAY, 1:00PM



Nothing is wrong.

@BatsouElef @eleftheriabatsou

THURSDAY, 2:00PM

Monitors		2:00 pm
Homepage	200	184 ms
API	200	96 ms
Login page	200	211 ms

“The checkout button doesn’t do anything.”

Support ticket #4812 · 2:00 pm

Probably their browser.

Still green.

@BatsouElef @eleftheriabatsou

THURSDAY, 2:30PM

Monitors		2:30 pm
Homepage	200	184 ms
API	200	96 ms
Login page	200	211 ms

Still green.

"Checkout button does nothing."

Ticket #4812 · 2:00 pm

"Can't complete my order."

Ticket #4817 · 2:20 pm

"Payment just spins."

Ticket #4819 · 2:20 pm

Sales: "Demo just failed in front of a prospect."

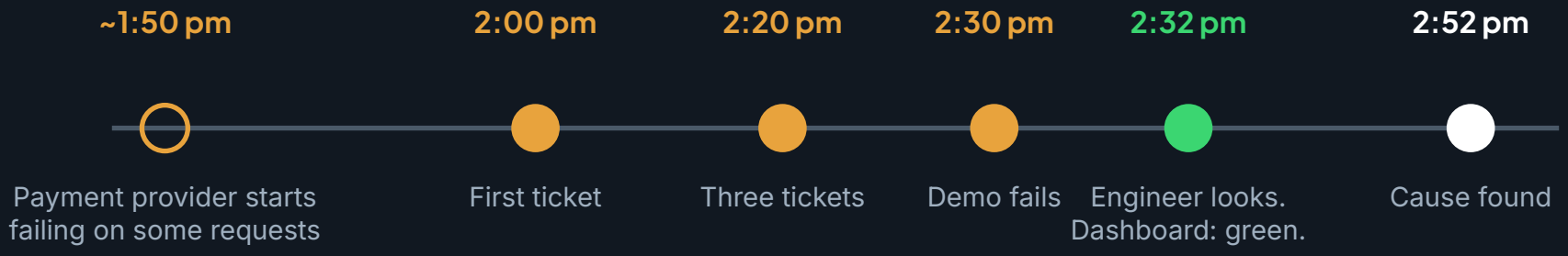
@BatsouElef @eleftheriabatsou

Monitors		2:32 pm
Homepage	200	184 ms
API	200	96 ms
Login page	200	211 ms



Every single check is passing.

WHAT ACTUALLY HAPPENED



Nothing you built was broken.

Your app was up for the entire incident.

~1 hour	1	some
of revenue	demo	trust

@BatsouElef @eleftheriabatsou

SWAP IN ANY DEPENDENCY



Payment provider

Nobody can pay



Identity provider

Nobody can log in



CDN

Half your assets don’t load



LLM API

Your headline feature returns nothing

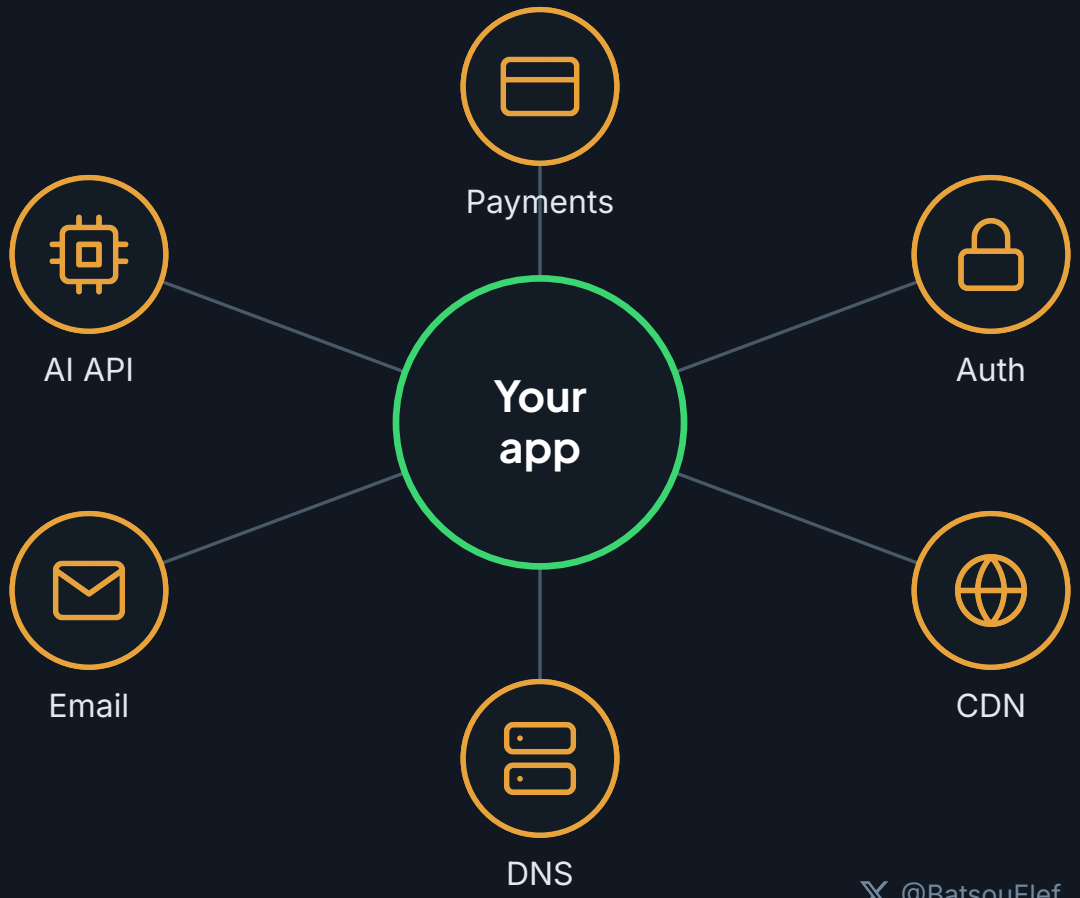
Same story. Different name.

@BatsouElef @eleftheriabatsou

PART TWO

The thing that broke isn’t the thing you built

THE SHAPE OF A MODERN APPLICATION



@BatsouElef @eleftheriabatsou

Your code

- Read it
- Test it
- Deploy it
- Roll it back

A dependency

- Observe it from the outside
- Can’t read it
- Can’t deploy it
- Can’t roll it back

You can’t fix Stripe. You can only find out fast.

A dependency failure looks like nothing, **until it looks like everything.**



A few percent of requests failing



Latency: 200 ms → 4 s



Valid response. Empty body.

The process is running. The front door opens. Your app is “up.”

@BatsouElef @eleftheriabatsou



It's measuring the thing you built.

The thing you built is fine.

@BatsouElef @eleftheriabatsou

NOT HYPOTHETICAL

99

services reported problems that pointed back to AWS



One cause, upstream, in something none of those teams run.

May 8, 2026 · attributions from services' own incident reports, not verified root causes
Source: UptimeRobot Knowledge Hub, Oct 2026

@BatsouElef @eleftheriabatsou

PART THREE

Why adding more checks doesn’t fix it

Before



After



If every new check still points at your own endpoints,

you get more of the same signal

THREE WAYS A HEALTHY CHECK MISSES A REAL FAILURE

1

The 200 that lies

2

The partial failure

3

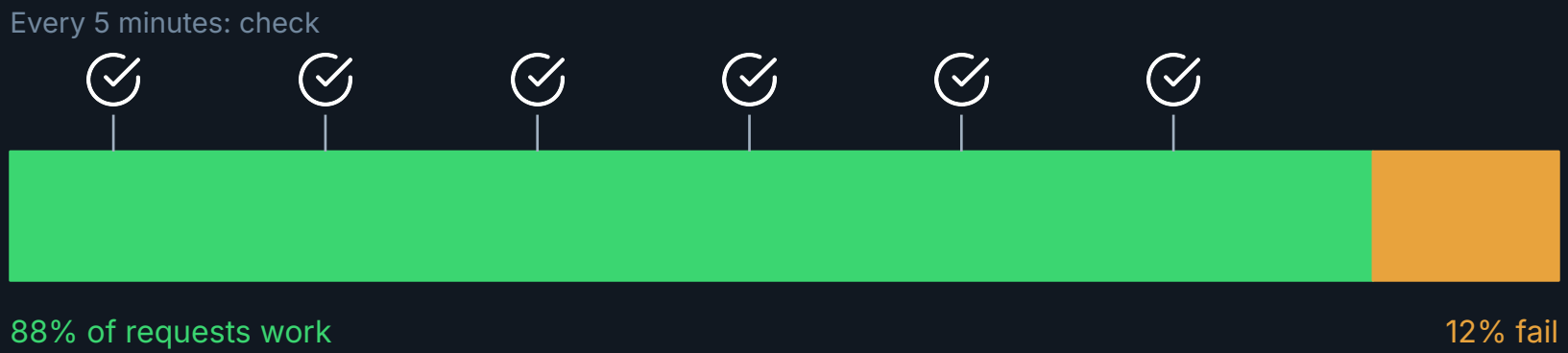
The slow success

200

The page rendered.

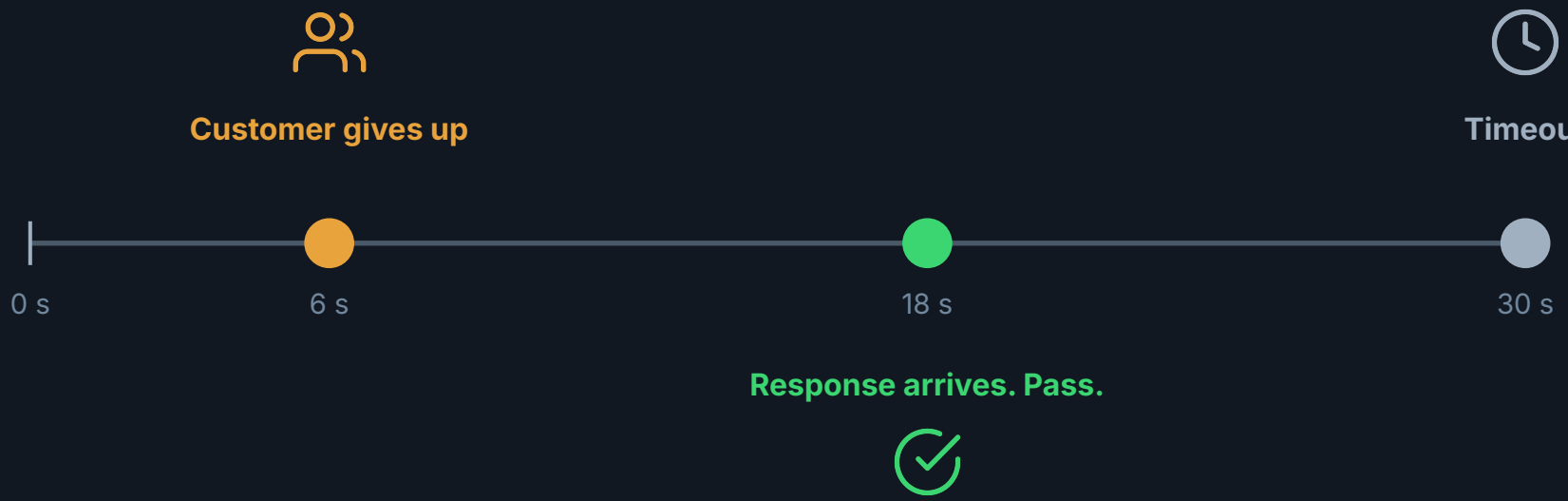
The payment didn’t.

A status code tells you the server answered. Not that the answer was right.



Six checks. Six passes.

A meaningful slice of your customers can’t buy anything.



Everything completes. Eventually.

Check what the customer does, not what the server says.

Plain HTTP check

"Did the server respond?"

Status code. Nothing else.

Keyword or API check

"Can a customer still do the thing they came for?"

Expected content. Real response. Human-scale latency.

Same URL. Completely different question.

@BatsouElef @eleftheriabatsou

QUICK DEMO

The 200 that lies, caught.

HTTP monitor

● Up

Checks the status code

Keyword monitor

● Down

Looks for the text a customer needs

Same URL. Same moment.

@BatsouElef @eleftheriabatsou

PART FOUR

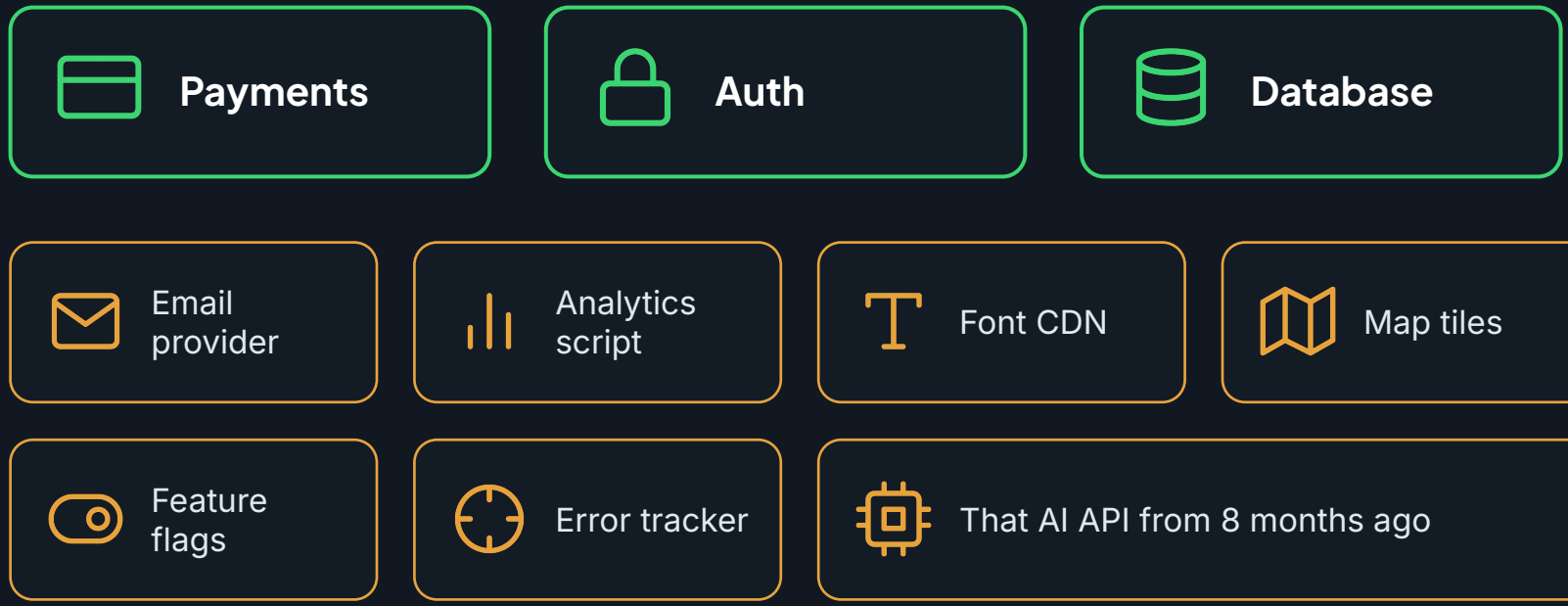
Know what you depend on

WHAT DOES YOUR APP DEPEND ON?



“That’s everything, right?”

WHAT DOES YOUR APP DEPEND ON?



Nobody has this list from memory.

@BatsouElef @eleftheriabatsou

WHERE DEPENDENCIES HIDE



Script tags and
third-party includes



Outbound calls
in your backend



DNS records,
especially CNAMEs



TLS certificates
issued to a vendor



Webhooks
you receive



Env vars full of
unaudited API keys

Two hours of unglamorous work. The highest-value task most teams never do.

@BatsouElef @eleftheriabatsou

GROUP THEM

Payments | Auth | CDN & DNS | Email | Analytics | AI

For each group, two questions:

1

If this degrades, what does a customer experience?

2

Would we know?

Honest answer, usually: "we'd find out from a support ticket."



Font CDN

Don’t need to



Payments

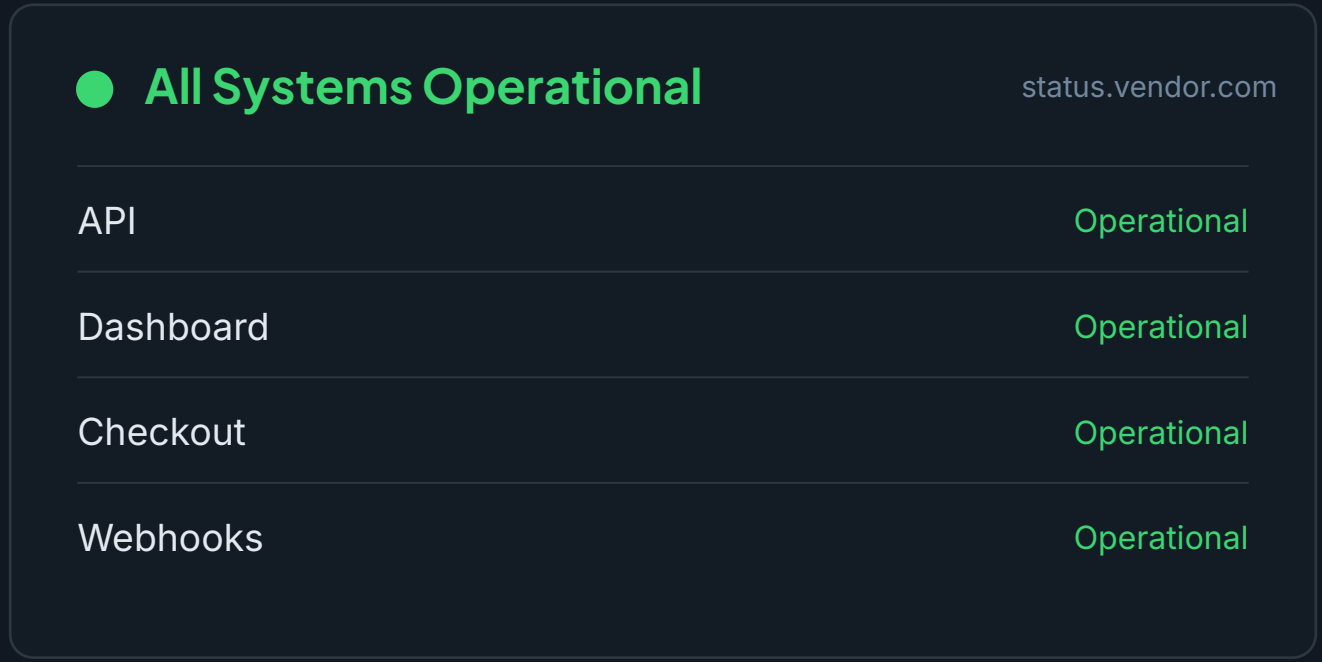
Absolutely need to

Monitor the thing that takes money.

PART FIVE

You can’t trust a platform to report its own outage

"IF STRIPE IS DOWN, STRIPE WILL TELL US."



Useful. Run in good faith, mostly. And not enough.

@BatsouElef @eleftheriabatsou

The party with the incentive to under-report is the party doing the reporting.

Structural, not moral.

@BatsouElef @eleftheriabatsou

SO STATUS PAGES...



Lag

Someone has to spot it,
confirm it and publish it first.



Gaps

Some incidents never reach
the status page at all.

WE MEASURED THE GAP

89%

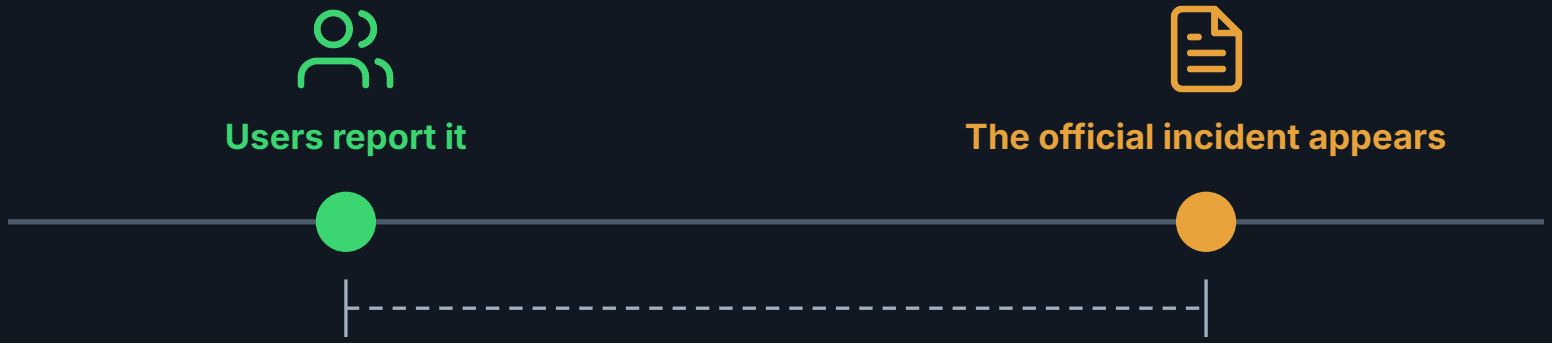
of crowd reports had no matching official incident within six hours

H1 2026 · services with an official status feed · UptimeRobot + IsDown data

Source: UptimeRobot Knowledge Hub, Oct 2026

@BatsouElef @eleftheriabatsou

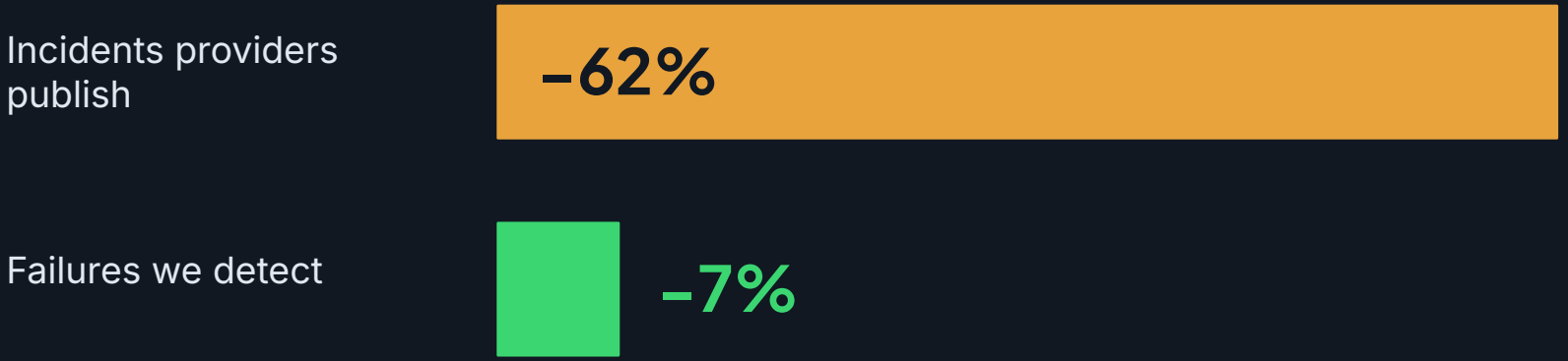
AND WHEN WE COULD MATCH THEM...



~2 hours

median lead

WHAT CHANGES ON WEEKENDS



The infrastructure doesn't know it's Saturday.

The people who write status pages do.

Weekend vs weekday · H1 2026 · UptimeRobot + IsDown data

@BatsouElef @eleftheriabatsou



Two hours is a long time when it’s your checkout.

@BatsouElef @eleftheriabatsou



The vendor's experience of the incident

That's what the status page reports.

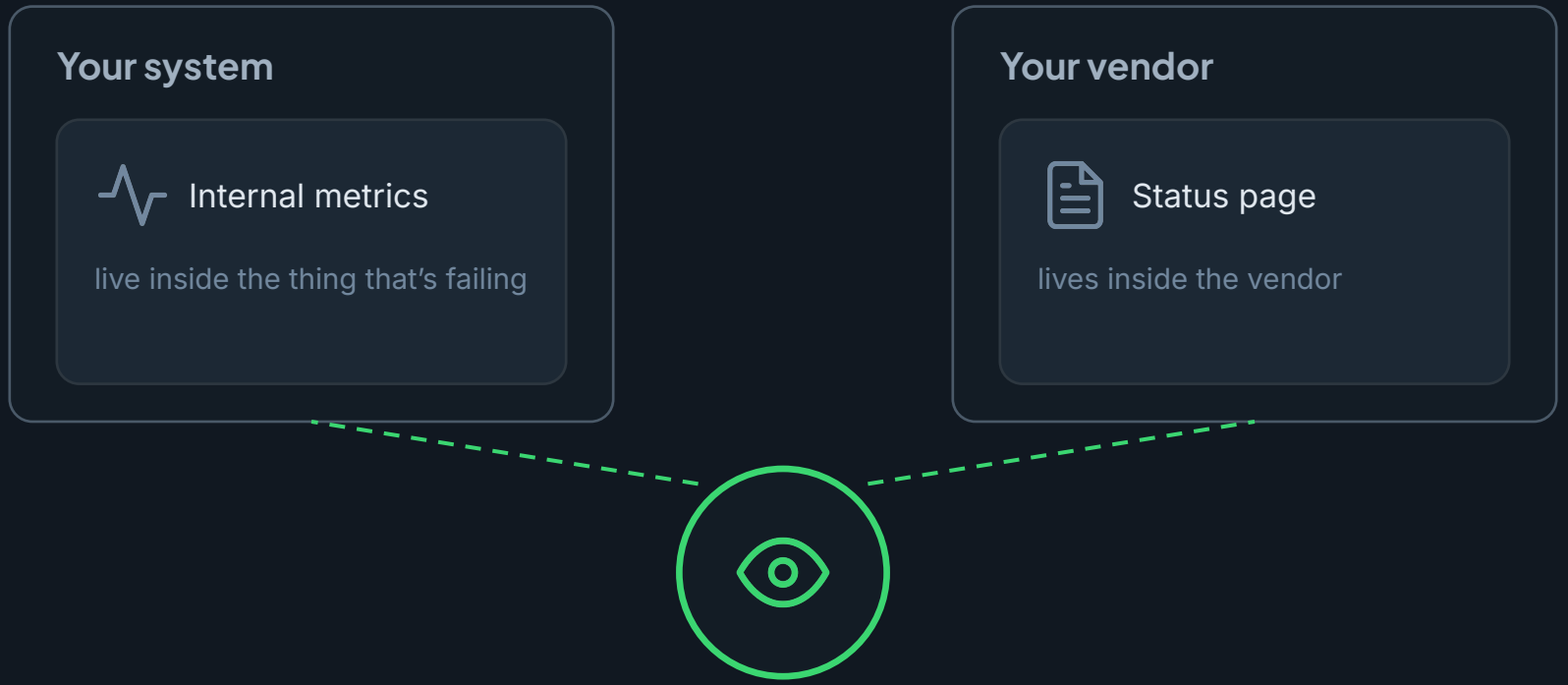


Your experience of the incident

That's what you need to know.

Those are different things.

THE OUTSIDE OBSERVER



Outside both. The same requests a customer makes, from where customers are.

A dashboard is a view from the inside.

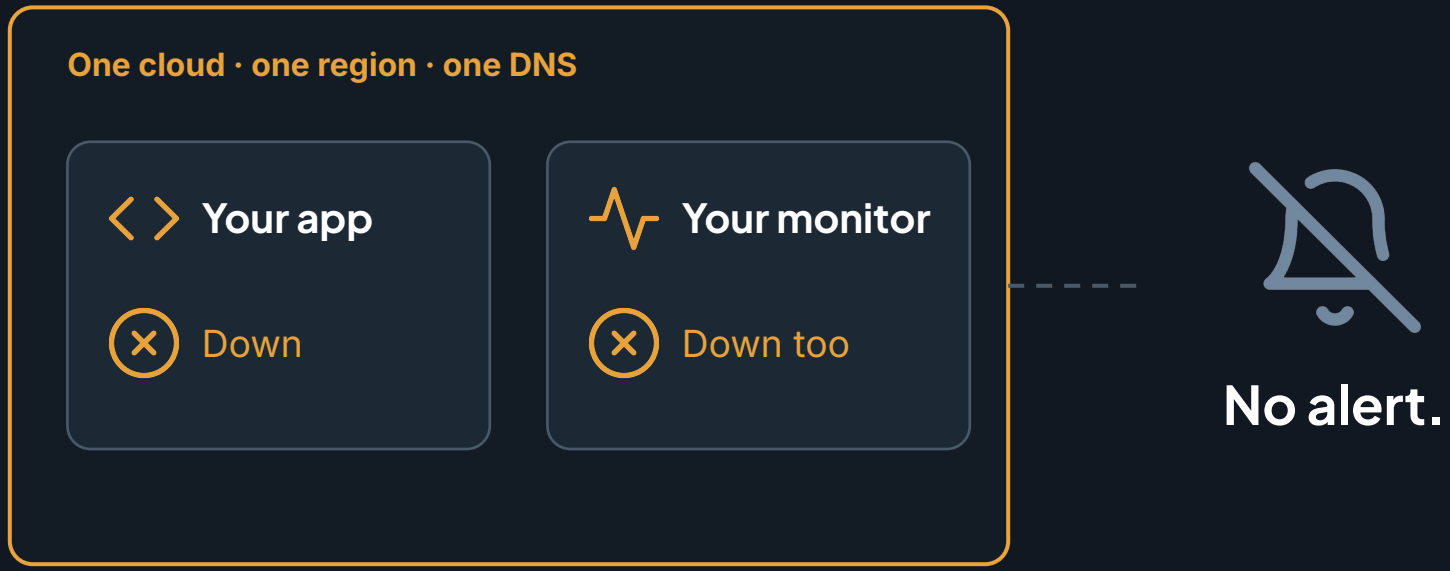
You need at least one view from the outside.



Why not just build it yourself?



SAME PLACE, SAME OUTAGE



If your checker sits in the same cloud, region and DNS as your app, the outage that takes your app down takes your checker down too



One location

“Is the site down, or is it me?”



Several locations

Confirmed before anyone gets paged.

That’s what stops you being woken up for nothing.



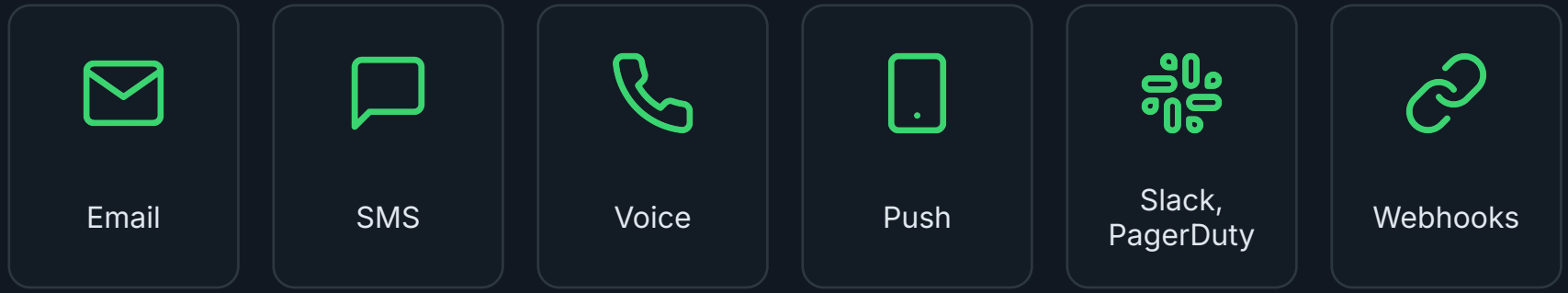
3:00 AM

Retries | Flapping | Timeouts | Maintenance windows | Confirmation

The hard part isn’t the request.

It’s deciding when to wake a human.

@BatsouElef @eleftheriabatsou



Each one is a small product of its own.

Each fails in its own way.



What you wanted

Ship your product.



What you also got

- Certificates
- Dependencies
- Storage
- The cron job that quietly stopped

You want to be the developer.

Not the person paged because the pager broke.

Build one to learn.

Don’t make it your only one.

If you self-host, use a different provider and region from your app.

@BatsouElef @eleftheriabatsou

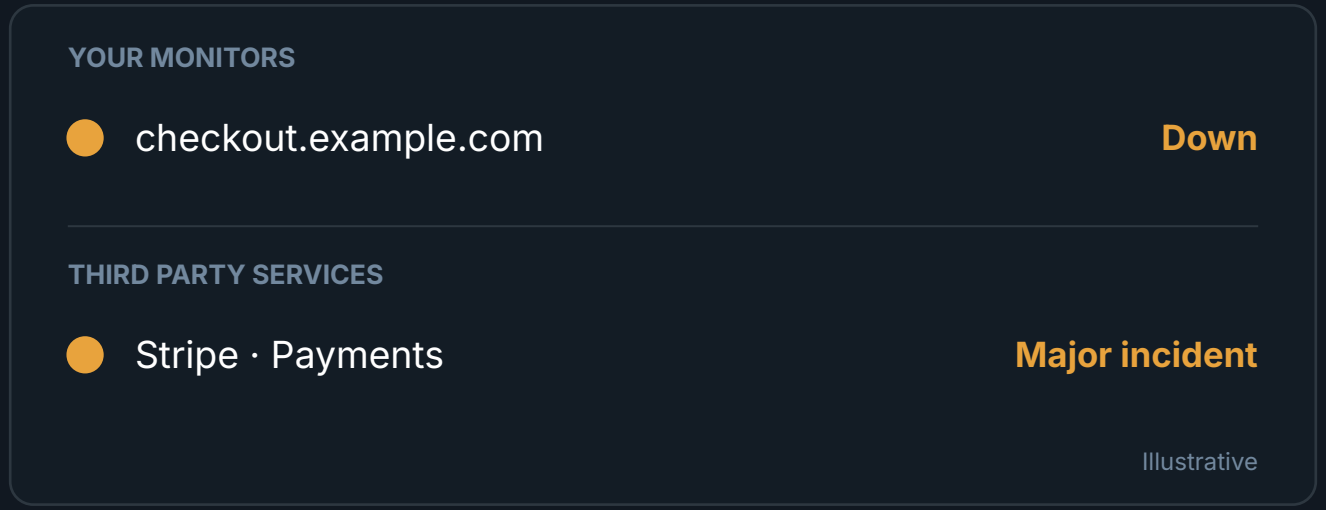
PART SEVEN

Seeing both sides of an outage



@BatsouElef @eleftheriabatsou

Is it your code, or is it them?



Both in one place. Now you know where to look.

@BatsouElef @eleftheriabatsou



What the provider says + What other users report + What your own checks see

Together, they tell you more than any one.

THIRD PARTY MONITORING: THREE STEPS

1



Search 6,000+ providers

Search “S3”, find AWS.

2



Use the contacts you have

Slack, Teams, PagerDuty, email, SMS, webhooks.

3



Pick the parts you use

Stripe payments and webhooks, not the dashboard.

Also in the API

@BatsouElef @eleftheriabatsou

QUICK DEMO

Third party monitoring.

1. Search a provider
2. Use existing contacts
3. Pick the components
4. Open a provider page

@BatsouElef @eleftheriabatsou

What to do this week

THIS WEEK

1. **Build the dependency list** — Homepage, key pages, backend calls, DNS, API keys. Two hours. Write it down.
2. Check the dependency itself
3. Upgrade “did it return 200”
4. Put the checks outside
5. Believe your checks

@BatsouElef @eleftheriabatsou

THIS WEEK

1. **Build the dependency list** — Homepage, key pages, backend calls, DNS, API keys. Two hours. Write it down.
2. **Check the dependency itself** — For every dependency where a failure costs money or locks people out, add a check, and track the provider's own incidents.
3. Upgrade “did it return 200”
4. Put the checks outside
5. Believe your checks

@BatsouElef @eleftheriabatsou

THIS WEEK

1. **Build the dependency list** — Homepage, key pages, backend calls, DNS, API keys. Two hours. Write it down.
2. **Check the dependency itself** — A check that calls it the way your app does, plus the provider's incidents next to your monitors.
3. **Upgrade "did it return 200"** — To "did it return what a customer needs." Keywords, response validation, human-scale latency.
4. **Put the checks outside**
5. **Believe your checks**

@BatsouElef @eleftheriabatsou

THIS WEEK

1	**Build the dependency list**	Homepage, key pages, backend calls, DNS, API keys. Two hours. Write it down.
2	**Check the dependency itself**	A check that calls it the way your app does, plus the provider's incidents next to your monitors.
3	**Upgrade "did it return 200"**	To "did it return what a customer needs." Keywords, response validation, human-scale latency.
4	**Put the checks outside**	Outside your infrastructure. Outside your vendors. Independence is the point.
5	**Believe your checks**	

@BatsouElef @eleftheriabatsou

THIS WEEK

1	**Build the dependency list**	Homepage, key pages, backend calls, DNS, API keys. Two hours. Write it down.
2	**Check the dependency itself**	A check that calls it the way your app does, plus the provider's incidents next to your monitors.
3	**Upgrade "did it return 200"**	To "did it return what a customer needs." Keywords, response validation, human-scale latency.
4	**Put the checks outside**	Outside your infrastructure. Outside your vendors. Independence is the point.
5	**Believe your checks**	When the status page says all clear and your checks disagree, your checks are right.

@BatsouElef @eleftheriabatsou

App up ≠ Business up

@BatsouElef @eleftheriabatsou

Try it on the free plan

50 monitors · no credit card

Third party monitoring is part of the paid plans.



Eleftheria Batsou

Developer Advocate, UptimeRobot

@BatsouElef · uptimerobot.com

Got a green-dashboard war story?
I'd love to hear it.

Questions?

Thank you!



Eleftheria Batsou

Developer Advocate, UptimeRobot

@BatsouElef · uptimerobot.com