

CASE STUDY · MONITORING + AUTO-HEALING

A watchdog that fixes what's safe to fix — and only calls a person for the rest.

Most monitoring just sends alerts. This watchdog checks a system around the clock, makes sure a failure is real, **repairs it by going back to the last known-good state**, proves the repair worked, and stays quiet unless a person is actually needed. The same pattern runs today on websites, contact forms and background jobs.

THE LOOP — ANY SYSTEM	EXAMPLE: A LIVE WEBSITE
1 Check it the way a user would On a schedule, around the clock — the real thing, not just "is the server on?"	Loads the real page every 2 minutes and checks it still names the business.
2 Make sure it's really broken One blip is never enough to act on.	Two failed checks in a row, about two minutes apart.
3 Rule out the watchdog itself If it can't see clearly, it touches nothing. "No answer" is never treated as "broken."	Asks three independent sites; two must answer. Many sites failing at once points at its own side.
4 Decide if it's safe to fix alone A change a person made is never overwritten — that gets an alert, not a repair.	The page loads but its wording changed: probably the client's own edit, so alert only.
5 Go back to the last known-good state Only ever a state recorded while everything was passing — never something newer.	Restores the last live version recorded while the site passed its checks.
6 Prove it worked Fixed: logged quietly. Not fixed: one text saying it needs a person.	Checks again right away. Still down: one text, at most every 30 minutes.

THE SLOW WAY

Something breaks at 2 a.m. A phone buzzes, someone logs in and does the same fix as last time — or the alert gets ignored, because the last ten were false alarms.

WHAT CHANGES

The routine fix happens by itself within minutes, and is checked. I only hear about what really needs a person, **so every alert means something.**

BY THE NUMBERS

2 min

between checks, around the clock

3 tries

then it stops and texts a person

6 a day

most repairs for any one system

0

texts when it fixes a problem by itself

WHERE IT RUNS TODAY

Websites

EVERY 2 MIN

Nine live client sites, checked, restored automatically when they break, and verified after.

Contact forms

EVERY 15 MIN

Every form is checked: it must send to the right place and never say "thank you" when the send failed. Lead emails are checked for delivery every hour. The checks never email anyone.

Background jobs

EVERY RUN

A crashed run's leftover lock is cleared, "busy" means try again later, a bad entry is set aside so it can't block the rest, and after the last retry I get one text.

WHAT KEEPS IT SAFE

✓ It never acts blind

A request that got no answer is not proof of failure. If the watchdog can't reach the internet, or many things fail at once, it touches nothing.

✓ It can't loop

Five minutes between repairs, three tries and then it stops and texts me, and at most six repairs a day for any one system.

✓ It only goes backwards

A repair returns to a state that was live and passing — never a draft, never something newer.

✓ It names the culprit

The alert says whether the provider reports an outage, the problem is on our side, or neither — instead of an hour of digging.

WHAT IT TELLS ME

Three possible outcomes, and only two reach my phone. A repair that worked is written to the log and nothing else. A repair that didn't work, or a problem it chose not to touch, sends one text saying exactly which.

```
# website example: the three lines it can report
client-site.com: auto-restored, verified back up
client-site.com: restored but STILL DOWN (...)
client-site.com: unreachable (...) — NOT restored

# the first is only logged; the others text me,
# at most once every 30 minutes per site
```

WHERE THE SAME IDEA FITS

Office systems

Any tool with a known-good state: check it, restore it, prove it, and page a person only for the rest.

Data jobs and syncs

Imports and syncs that retry, set bad records aside and say exactly what failed — instead of failing quietly.

Agencies

Client sites and forms that fix themselves before the client notices.

Built with Python · scheduled jobs on a Linux server · the hosting provider's API · the email provider's delivery API · public status pages · SMS alerts. Every safety rule has a test that fails if the rule is removed.

Want monitoring that fixes things, not just alerts about them?

I build watchdogs that repair what's safe to repair and only call a person for the rest.

linkedin.com/in/julianholien