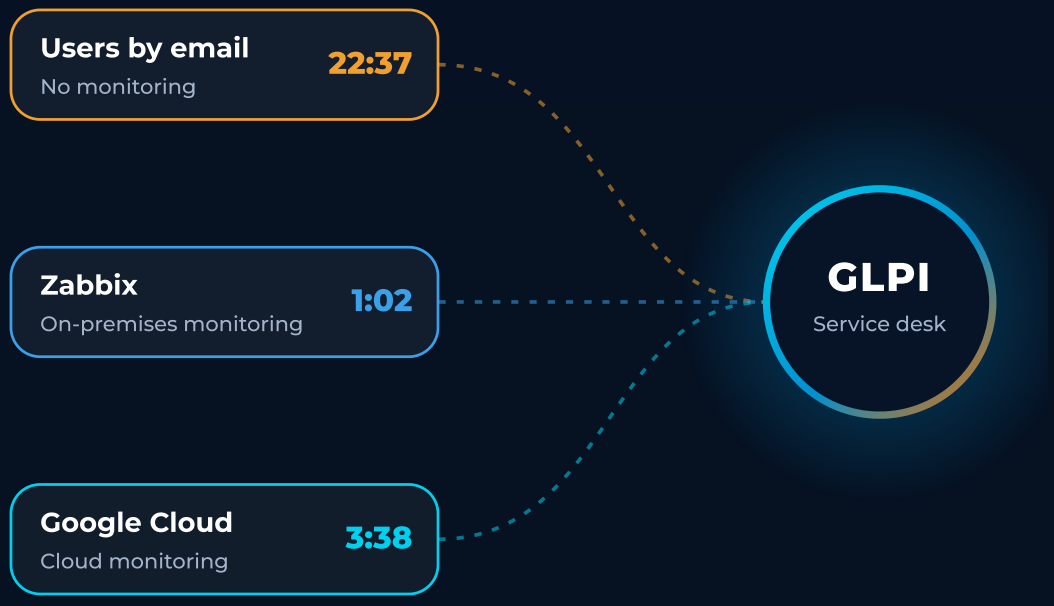
MARIFEX INSIGHTS OUTAGE DETECTION

Who tells the service desk first?

How fast does an outage reach GLPI? The same ERP failure timed with no monitoring (22:37), Zabbix (1:02) and Google Cloud (3:38), and the tickets each path left.

MarifeX GLPI practice 17 Oct 2026 9 min read



Minutes from the ERP failure until the service desk knows

EXECUTIVE SUMMARY

The situation

An ERP front end fails. Three organisations run GLPI, but only two of them watch the ERP, and each one finds out in a different way.

Our approach

We ran the same failure in real time on each path, one run per path, and recorded what reached GLPI, when, and in what shape.

The outcome

Detection ranged from one minute to over 22. The faster paths were not automatically the better ones; how the record was structured mattered as much as the speed.

22:37

minutes until the desk knew, with no monitoring

1:02

minutes with Zabbix

3:38

minutes with Google Cloud

3 vs 1

tickets for one unstable service, Zabbix webhook against a custom cloud function

01 The situation

The time between a failure and the moment the service desk knows about it is downtime that nobody is working on.

An ERP front end used by finance, sales and the warehouse fails mid-morning. Three organisations run GLPI, and each finds out differently:

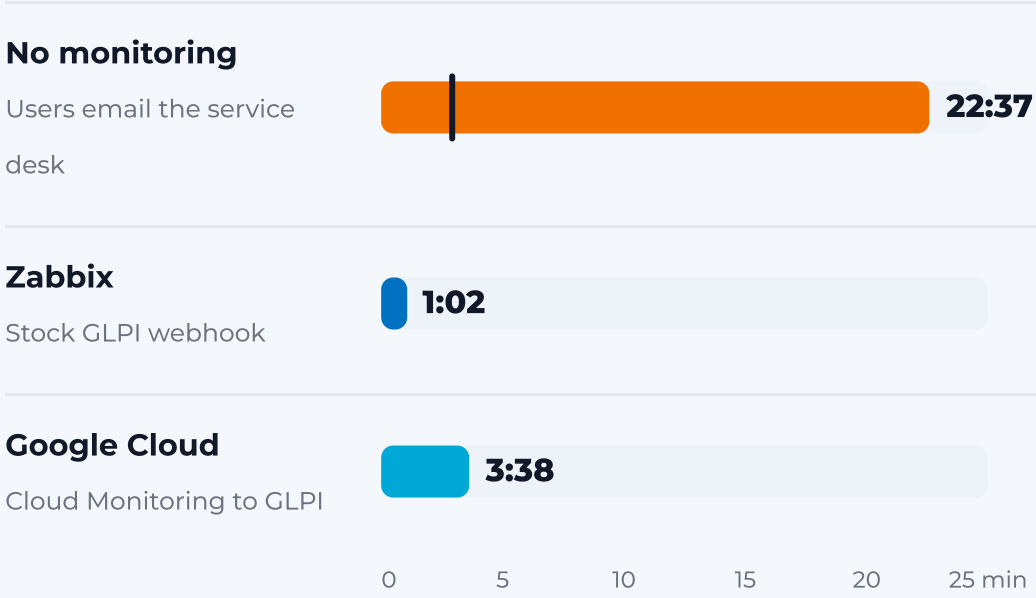
- **No monitoring:** people notice and email the service desk.
- **Zabbix** on premises, with its GLPI webhook switched on.
- **Google Cloud**, where Cloud Monitoring watches the ERP.

We ran the same failure for each of the three in its own real-time run and timed each step until the service desk knew.

EXHIBIT 1

Minutes from the ERP failure until the service desk knows

The same failure, one real-time run per organisation



First sign of trouble: the first user email, 2 min 50 s after the failure. It became a ticket 11 min 18 s after the failure.

Source: MarifeX reproduction of the scenario on standard GLPI.

02 The challenge

Detection speed is only part of the problem. What reaches GLPI must also be a record the service desk can act on.

01

A late signal

Without monitoring, detection depends on user reports, and GLPI sees them only when its scheduled tasks run.

02

A silent alert

An alert that fires but reaches nobody is worse than none, because everyone assumes it works.

03

Ticket noise

A service that fails, recovers and fails again can open a new ticket every time.

04

Records that close themselves

A ticket solved by a recovery signal says the check passed, not that users can work.

> **Speed only helps if what reaches GLPI is one record the service desk can own.**

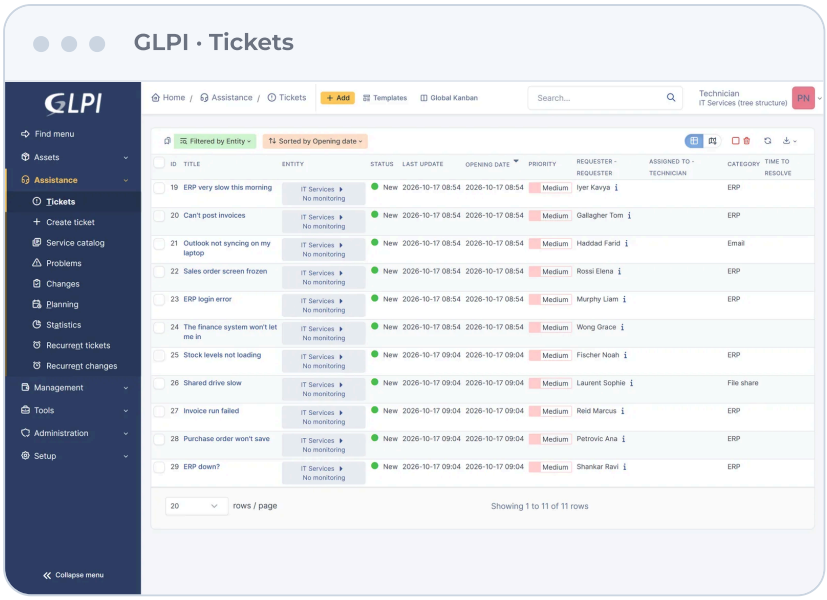
03 How each organisation finds out

Times are counted from the moment the ERP failed.

NO MONITORING Detection depends on user reports

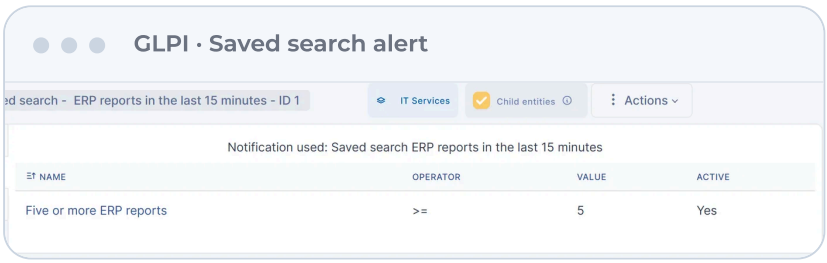
The first user email arrived 2 minutes 50 seconds after the failure. Eleven reports followed over the next 17 minutes, nine of them about the ERP.

The mail collector converts the emails into tickets, and routing rules set the category from keywords in the message. The collector runs every ten minutes by default, so the reports reached GLPI in two batches, 11 and 20 minutes after the failure.



Eleven user reports reached GLPI in two batches, ten minutes apart. One ERP report was left without a category.

A saved search counts open ERP tickets from the last 15 minutes and alerts the service lead at five or more. That check also runs on a schedule. The lead was alerted after 22 minutes 37 seconds.



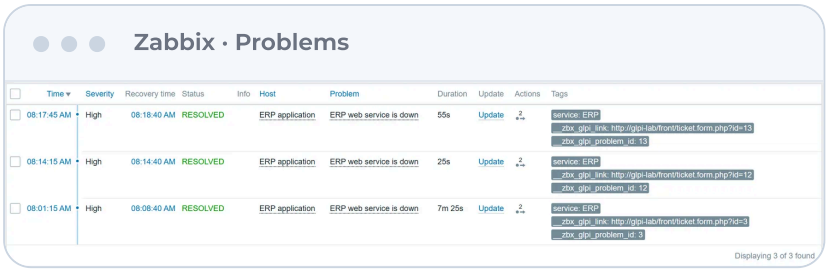
The saved-search alert: five or more open ERP tickets within 15 minutes.

Two configuration gaps affected the result. One ERP report used none of the keywords in the routing rule, so it was not categorised and did not count towards the alert. On the first attempt the alert fired but **nobody received it**: for a shared saved search, GLPI sends the alert only once a notification exists for that search, and the button that creates it leaves it inactive.

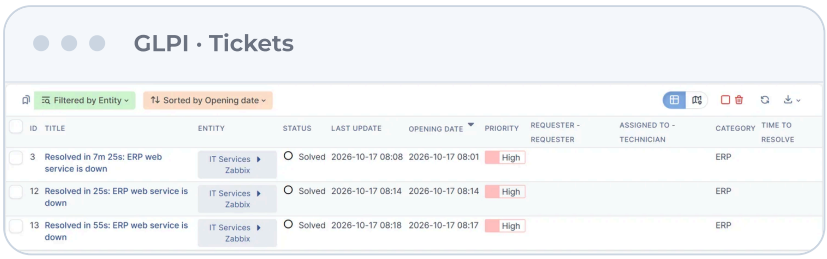
ZABBIX The alert is the first report

Zabbix checks the ERP every 30 seconds and raises a problem after two failed checks. Its GLPI webhook opens a ticket through the GLPI API. The ticket was in GLPI after **1 minute 2 seconds**, almost two minutes before the first user email.

When the ERP came back eight minutes later, the webhook solved the ticket within 27 seconds and renamed it “Resolved in 7m 25s: ERP web service is down”. Nobody at the service desk had touched it. Two more short failures, a little over a minute each, opened two more tickets, each solved automatically when the check passed.

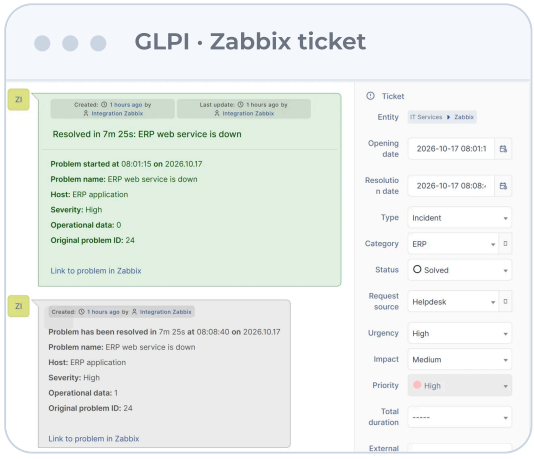


Zabbix: three problems for one unstable service, each with its own GLPI ticket.



GLPI: three tickets, each solved automatically by the webhook.

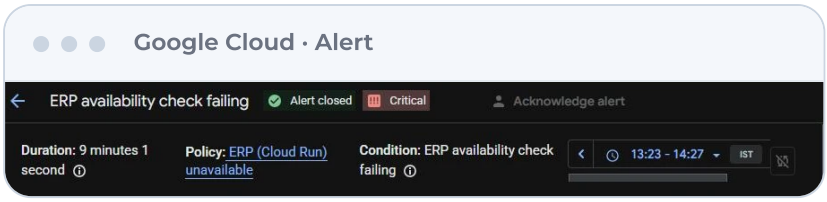
The webhook works as written: one ticket per Zabbix event, solved on recovery. For the service desk that means three records for one service, no link to the ERP service record, and tickets closed before anyone at the service desk reviewed them. A GLPI ticket rule can add the service link; we did not set one up for this run.



The integration user opened and solved the ticket; no agent was involved.

GOOGLE CLOUD Alerts reach GLPI on the internal network

Here the ERP front end runs on Cloud Run. Cloud Monitoring checks it every minute from three regions using authenticated requests, so the ERP does not need to accept anonymous traffic. The alert fires when more than one region fails for a minute.



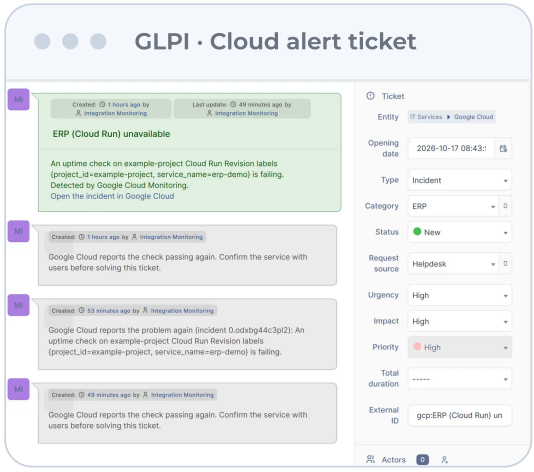
The Cloud Monitoring alert for the first failure: critical, closed after 9 minutes 1 second.

The alert goes to Pub/Sub, which passes it to a small function that only accepts Google-signed requests. The function reaches GLPI over the internal network, so GLPI is not exposed either. The ticket was in GLPI after **3 minutes 38 seconds**, linked to the “ERP (Cloud Run)” service record.

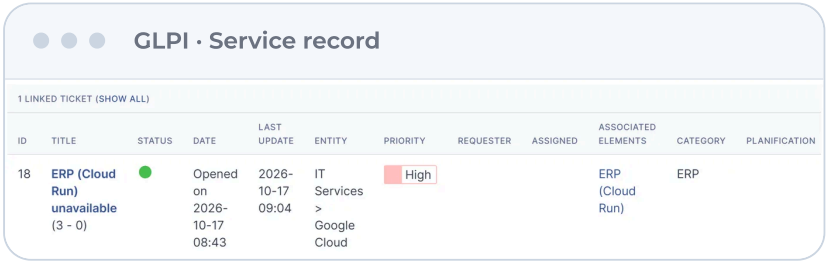
Cloud Run can also be set to internal-only access, which Google documents as still accepting uptime checks and Pub/Sub from the same project. That would take both services off the internet; we did not test that setup.

We designed the function around the service desk:

- **One ticket per affected service** while that ticket is open.
- **Every later notice is a follow-up**: the recovery, the second failure seven minutes later and the second recovery.
- **No automatic solving.** The last follow-up asks the desk to confirm the service with users.



One ticket covers the whole episode and remains open for the service desk.



The ticket is linked to the business service record, not to an individual server.

AWS and Azure offer the same building blocks: a CloudWatch alarm with SNS and Lambda, or an Azure Monitor alert with an action group and a function. We verified the pattern on Google Cloud only.

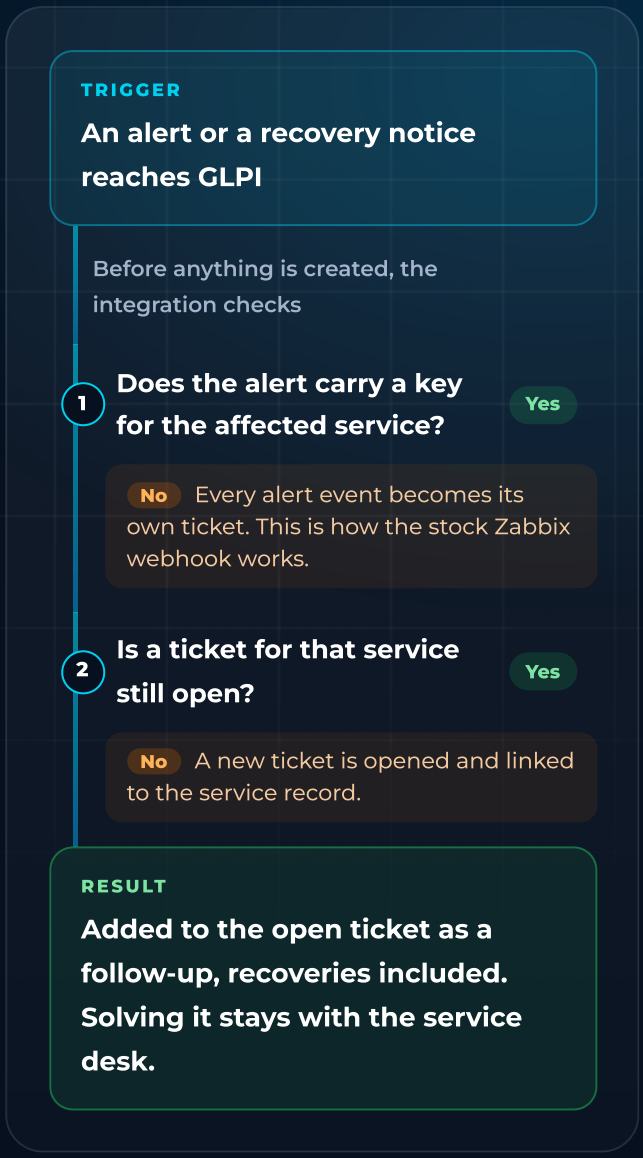
UNDER THE HOOD

One ticket or three: what decides it

Monitoring tools report events. A service desk manages services. The integration decides which of the two GLPI records.

In practice: key alert tickets on the **affected service**, add repeats and recoveries as follow-ups, and leave **solving to a person** who can confirm the service with users.

Behaviour of Zabbix's GLPI webhook and a custom Google Cloud function, verified on GLPI 11.0, Zabbix 7.0 and Google Cloud, October 2026.



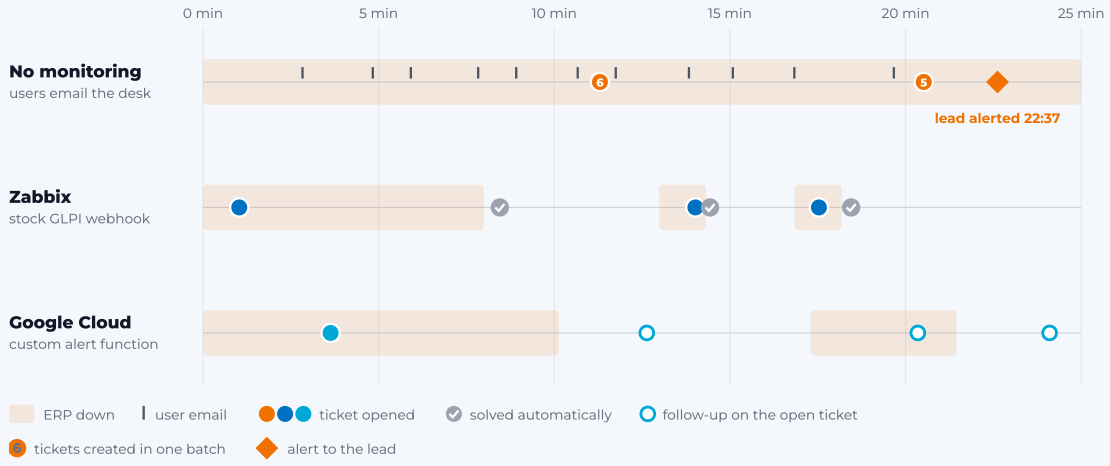
04 The outcome

Side by side, the paths show a trade-off between detection speed and how the record is structured.

EXHIBIT 2

The first 25 minutes on each path, record by record

Shaded bands are when the ERP was down in each run



Source: MarifeX reproduction of the scenario on standard GLPI.

EXHIBIT 3

What each path left in GLPI

Same failure and GLPI, one real-time run per path

	NO MONITORING	ZABBIX, STOCK WEBHOOK	GOOGLE CLOUD, CUSTOM FUNCTION
Failure until the service desk knows	22 min 37 s	**1 min 02 s**	3 min 38 s
Records for one failing service	9 ERP tickets, one per report	3 tickets	**1 ticket, 3 follow-ups**
ERP reports left uncategorised	1 of 9 ERP reports	none	none
Linked to the affected service	no; a ticket rule could add it	no; a ticket rule could add it	**yes, the ERP service record**
When the service comes back	nothing changes	ticket solved within 27 s, unattended	**follow-up only, ticket stays open**

Source: MarifeX reproduction of the scenario on standard GLPI.

- **Most of the 22 minutes came from configuration.** The first email arrived within three minutes; the scheduled mail collection and alert checks added most of the rest, and both intervals can be shortened.
- **The fastest path made the most noise.** Zabbix detected the failure in a minute but left three tickets, all closed automatically.
- **The cloud path was slower but more useful.** One ticket, linked to the service, covered the whole episode and stayed open.

- **No plugins were required.** Every path used standard GLPI, its REST API and the monitoring each organisation already had.

05 What we recommend

1. **Measure your own detection time**
   Take a test service down and time it until someone at the service desk knows.

2. **Without monitoring, shorten the schedules**
   Run the mail collector every minute or two, not every ten, and check alerts on the same rhythm.

3. **Let users name the service**
   A short form with a service field routes reports better than guessing from the words in an email.

4. **Test every alert end to end**
   For a shared saved search, create its notification, switch it on, give it recipients, and check that a message arrives.

5. **Key monitoring tickets on the service**
   Repeats and recoveries belong in the open ticket as follow-ups. With Zabbix, this means extending the stock webhook.

6. **Leave solving to people**
   A passing check is a follow-up, not a solution. Then run the incident as in our use case "One outage, forty tickets, one response".

HOW MARIFEX HELPS

From standard GLPI to the way your teams work

Most organisations do not need a different ITSM tool. They need the one they have to fit their process. Because GLPI is open source, we shape it around you instead of shaping you around a licence.

Adoption

We map your incident, problem and change process to GLPI and set up rules, SLAs, notifications and templates so the right steps happen without manual effort.

Customisation

We build plugins and workflow extensions where the standard product stops short, and we contribute fixes back to GLPI itself.

Integration

We connect monitoring, directories and reporting so incidents like this one are detected and measured, not only logged.

No lock-in

You own the platform, the data and every change we make. No per-agent licence tiers decide which features you can use.

Talk to our GLPI team: **marifextech.com/contact** info@marifextech.com

ABOUT THIS USE CASE

This use case shows the approach MarifeX follows when helping organisations adopt and run GLPI. The scenario is representative of what IT teams face, and it was rebuilt on a standard GLPI installation so that every screen and figure can be shown openly. The behaviour described was verified on GLPI 11.0, Zabbix 7.0 and Google Cloud, October 2026 and may change in later releases.

TRADEMARKS AND RIGHT TO USE

GLPI is a trademark of Teclib'. MarifeX is an independent GLPI service provider.

© 2026 MarifeX Technologies. You may share this document unchanged with attribution. You may not modify, sell or rebrand it. Licensed under CC BY-ND 4.0.