IT director / CTO



Your standards applied to every product, at a cost you control, where you choose to run it.

01 · ENGINEERING DIRECTOR

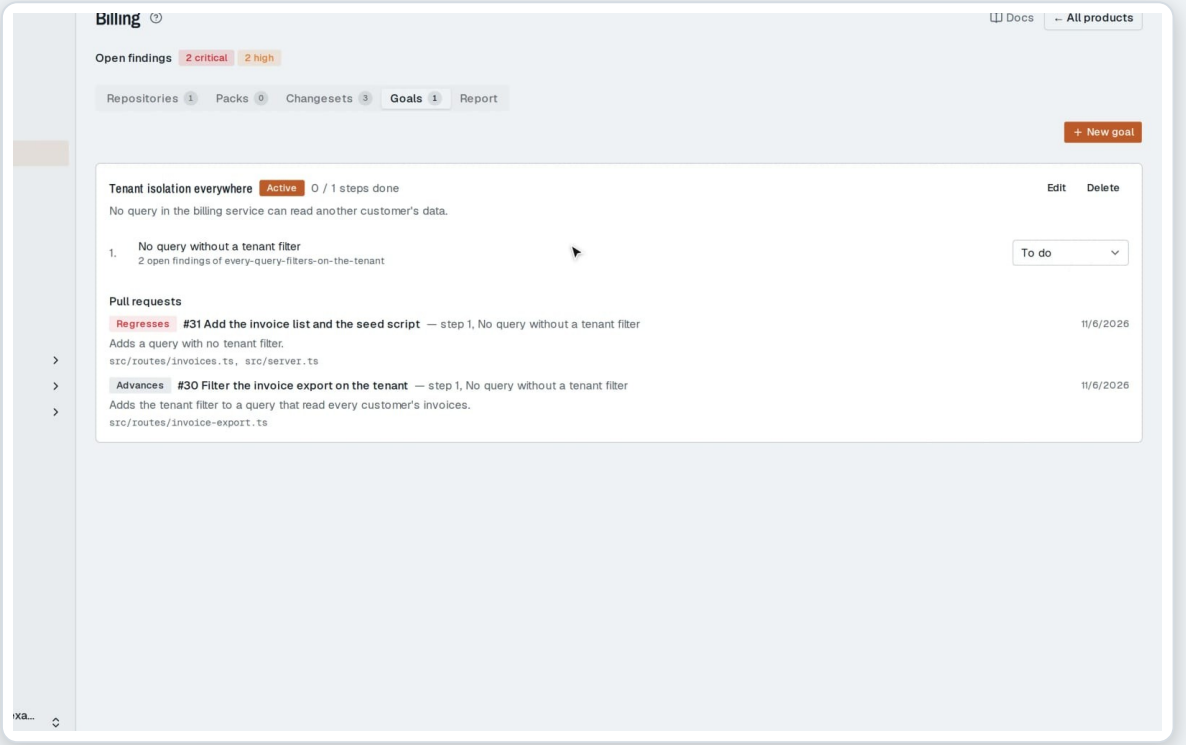
Track an engineering goal across pull requests

“As an engineering director, I want to know if our changes move us toward our goals, not only if they fix single issues.”

Goals like customer data isolation or a migration live in a slide deck. Nobody knows the real progress, and nobody sees when a pull request makes it worse.

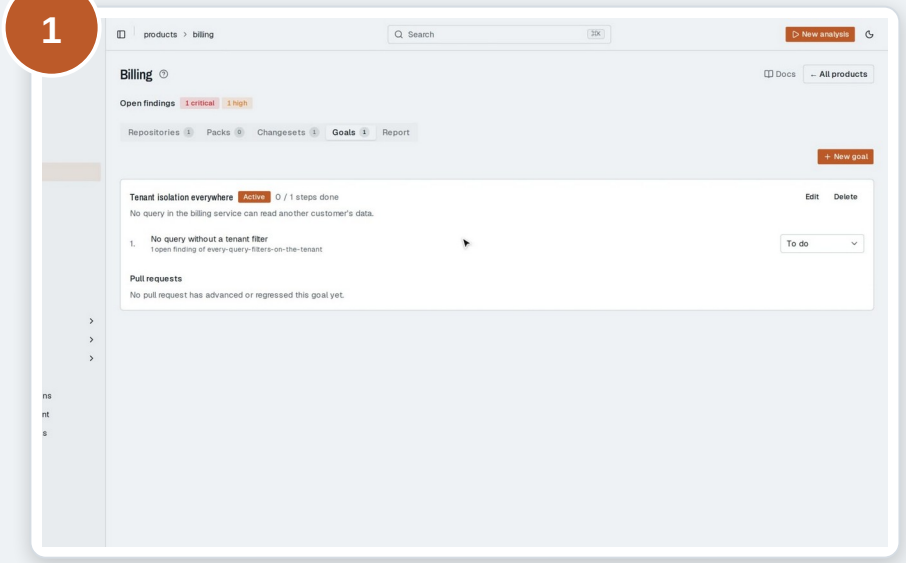
What to notice

- A measured step is done when its rules have no open findings left in the product.
- One pull request never completes the goal by itself. A step is done when its count reaches zero, or when a person marks it as done.



Track an engineering goal across pull requests

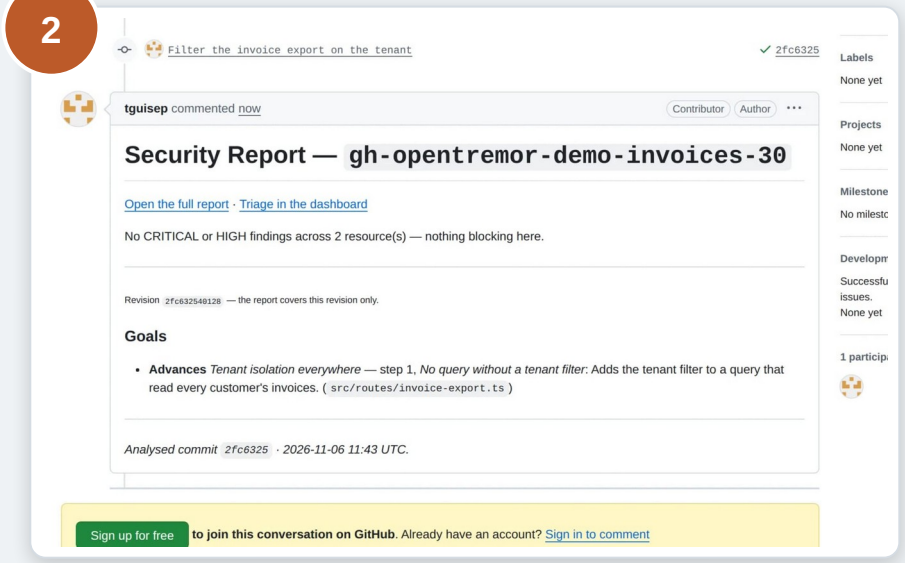
1



An engineering director sets a goal, with a step OpenTremor can measure.

The goal "Tenant isolation everywhere" has a step that OpenTremor measures: the number of open findings for one rule. A scan found one.

2

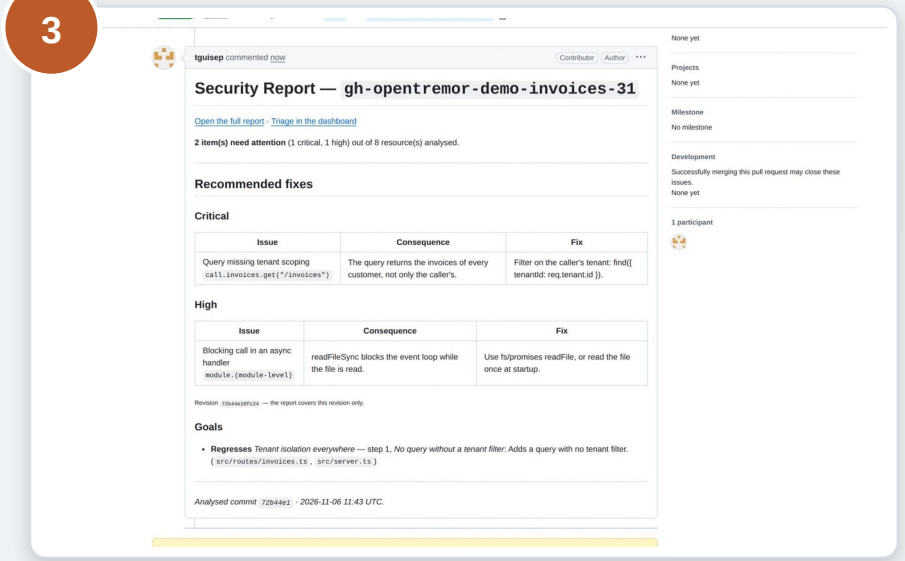


A pull request that fixes old code advances the goal.

A pull request fixes the old export. OpenTremor's comment says that it moves the goal forward, and explains why.

Track an engineering goal across pull requests

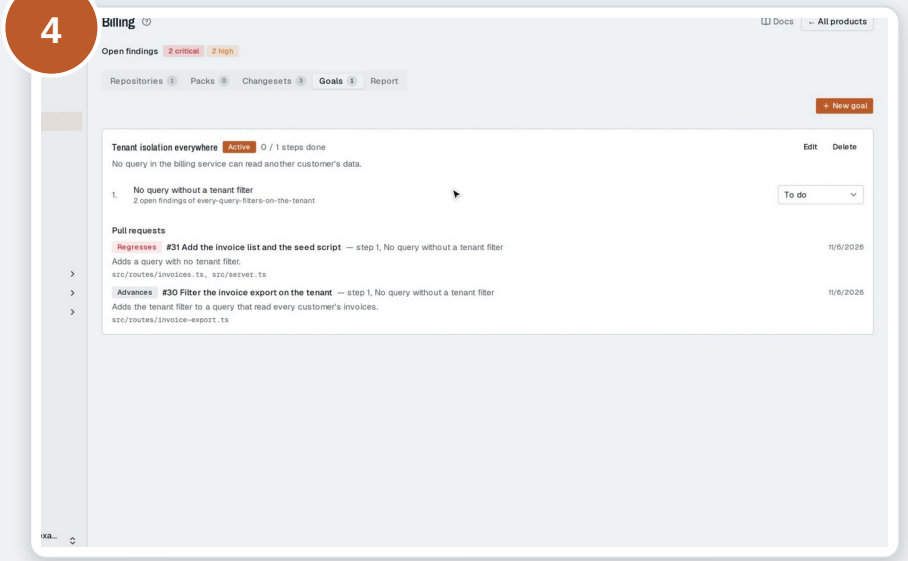
3



One that adds an unfiltered query sets it back, and the comment says so.

Another pull request adds a query that reads every customer's data. The comment says that it moves the goal backward.

4



The goal keeps the record: which pull request moved it, and why.

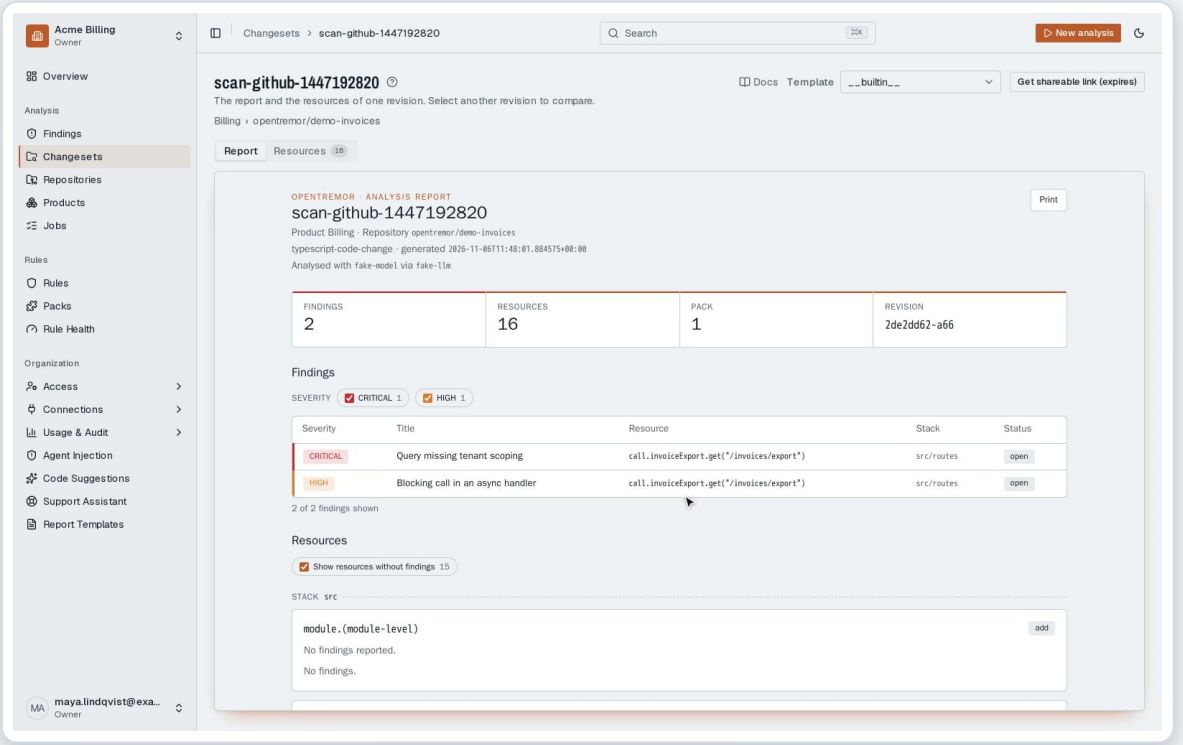
The goal keeps both pull requests, with the reason and the files, for the next review meeting.

02 · PLATFORM ENGINEER

Find problems already in your code

“As a platform engineer, I want to check an existing repository against our standards, so I can find its problems before they cause an incident.”

Pull request review sees only the changed code. Nobody checks the older code, so nobody knows how much of it breaks the rules.

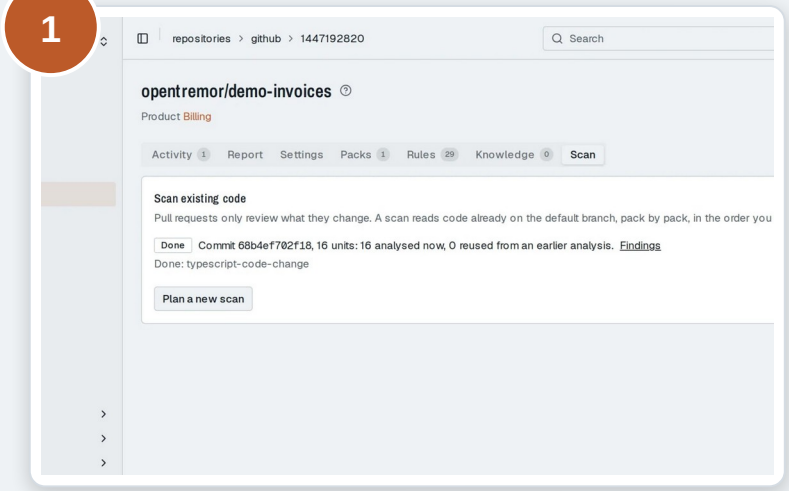


What to notice

- A scan never blocks a pull request. It only shows the current state of the code.
- OpenTremor never sends the same code to the AI model twice.

Find problems already in your code

1



A platform engineer scans the code already on the default branch.

The repository's Scan tab shows what a scan will check. One click scans the main branch.

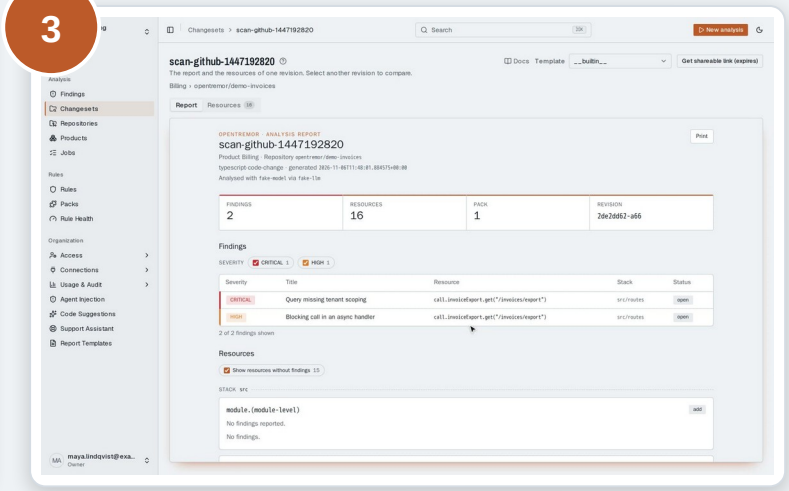
2



OpenTremor finds what the existing code breaks, rule by rule.

The scan finds an old CSV export that returns the invoices of all customers. Each finding names the rule and the risk.

3



The scan's report sorts the findings by severity: a backlog to work through.

The scan report counts the findings by severity: one critical, one high. The team can plan the work and assign it.

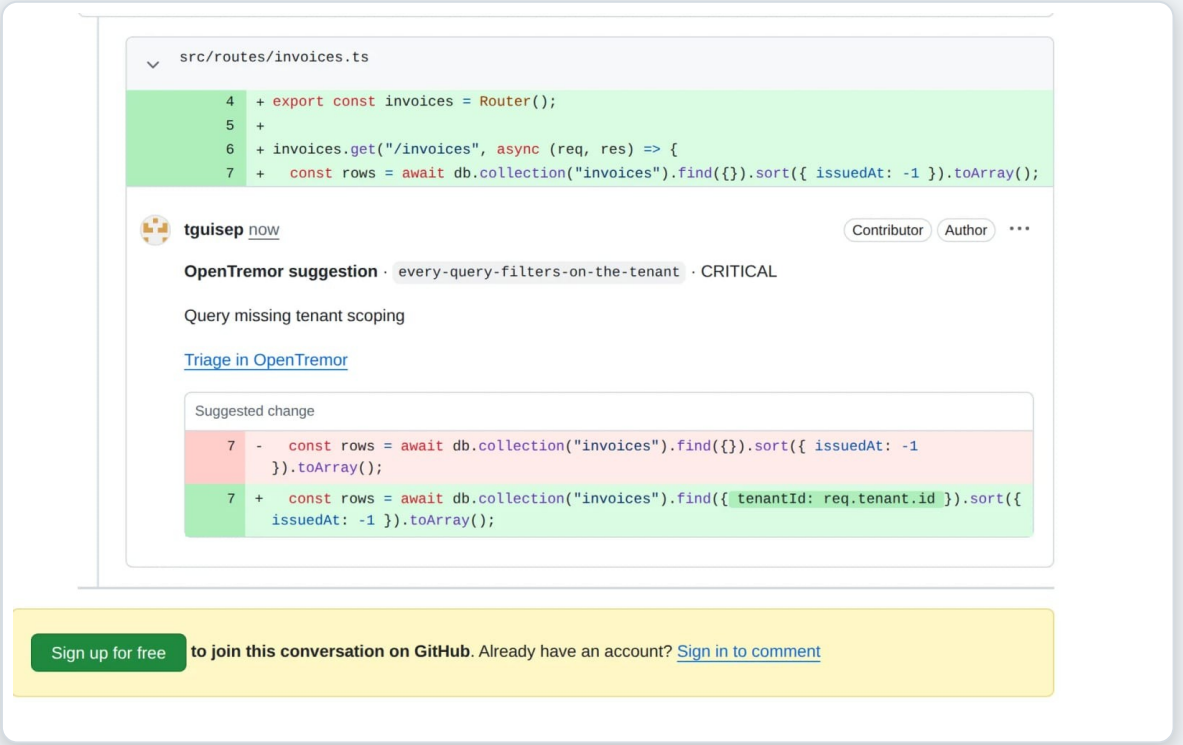
Every pull request checked against your team's rules

“As a tech lead, I want every pull request checked against our written standards, so reviewers stop repeating the same comments.”

Your standards are in a wiki and in people's heads. A pull request follows them only if the reviewer knows them and has time.

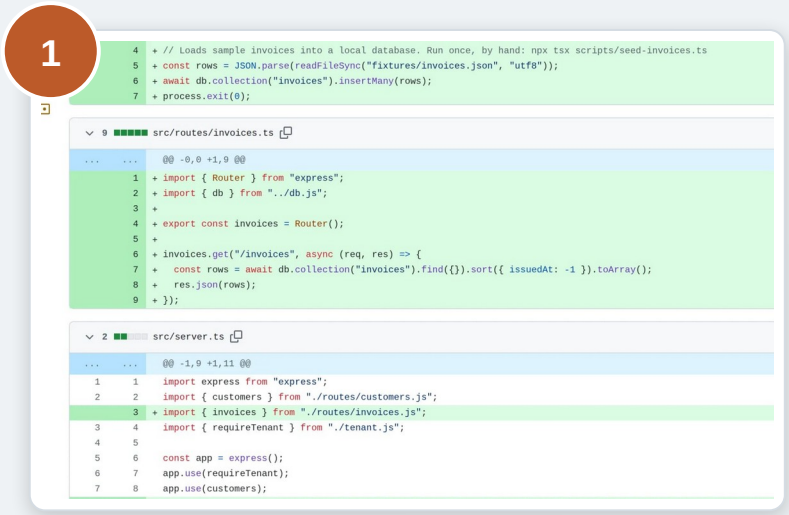
What to notice

- The rule is your team's own, written in plain words. You do not need a special rule language.
- The finding explains the risk, not only the line of code.
- The fix is a normal GitHub suggestion. OpenTremor never commits code for you.



Every pull request checked against your team's rules

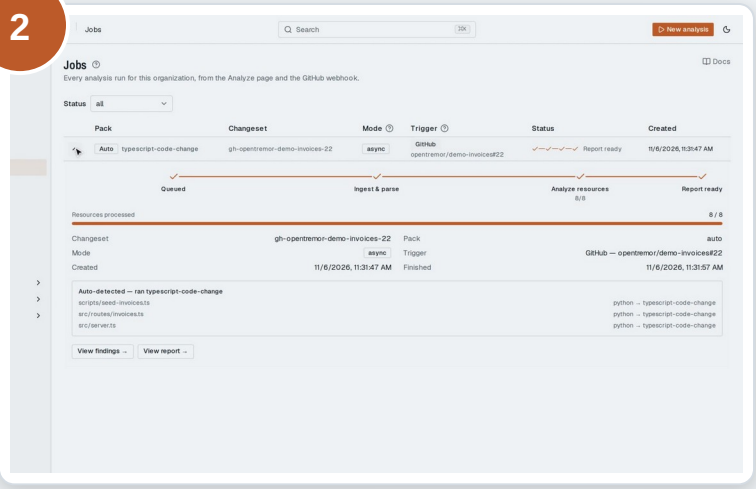
1



A developer opens a pull request.

The pull request adds a list of invoices. Its database query returns the invoices of all customers, not only the caller's.

2



OpenTremor reviews it against your team's rules.

OpenTremor receives the pull request and checks the changed code against the organization's rules.

3

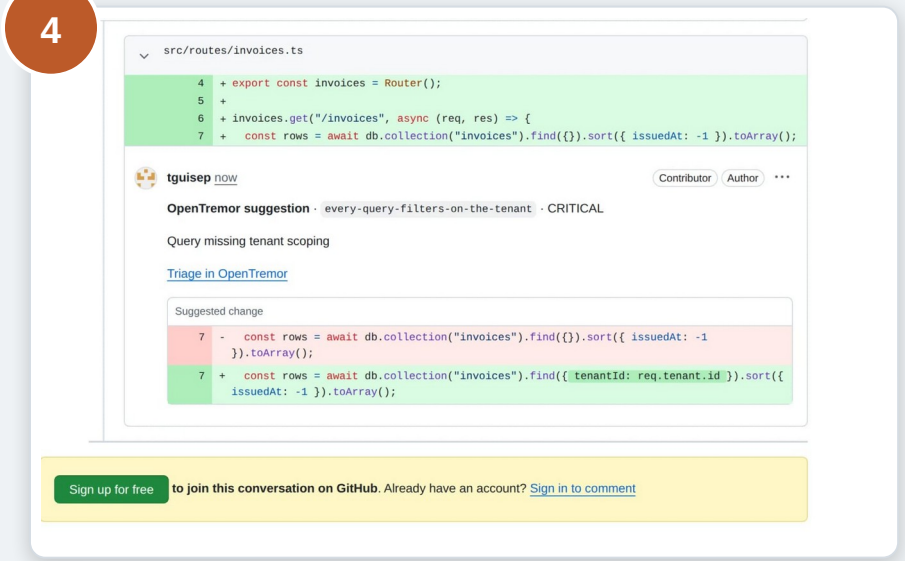


Each finding names the rule it breaks and why it matters.

The finding names the broken rule, "Every query filters on the tenant". It also explains the risk if the code goes to production.

Every pull request checked against your team's rules

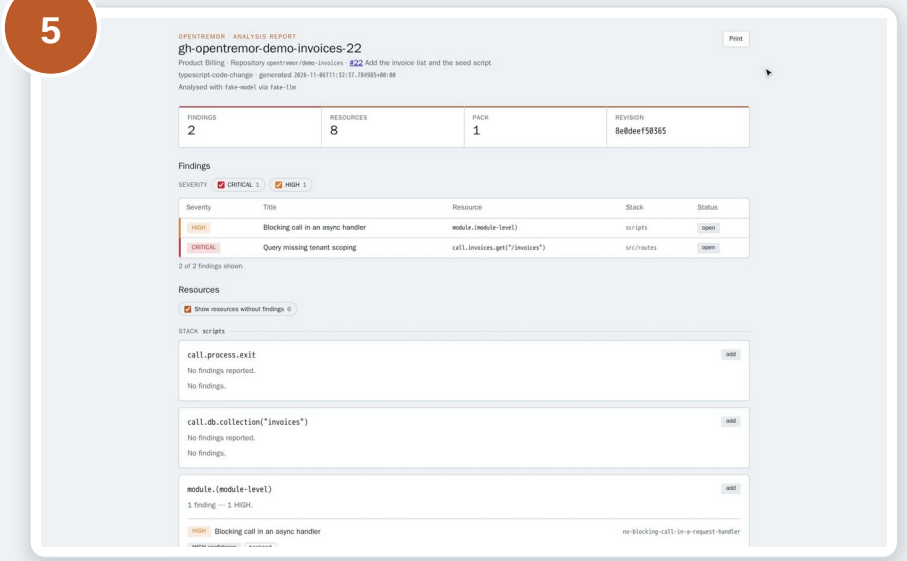
4



The fix comes back to the pull request, ready to commit.

For a critical finding, OpenTremor suggests the fix on GitHub. The developer accepts it from the pull request.

5



The full report, shared with one link.

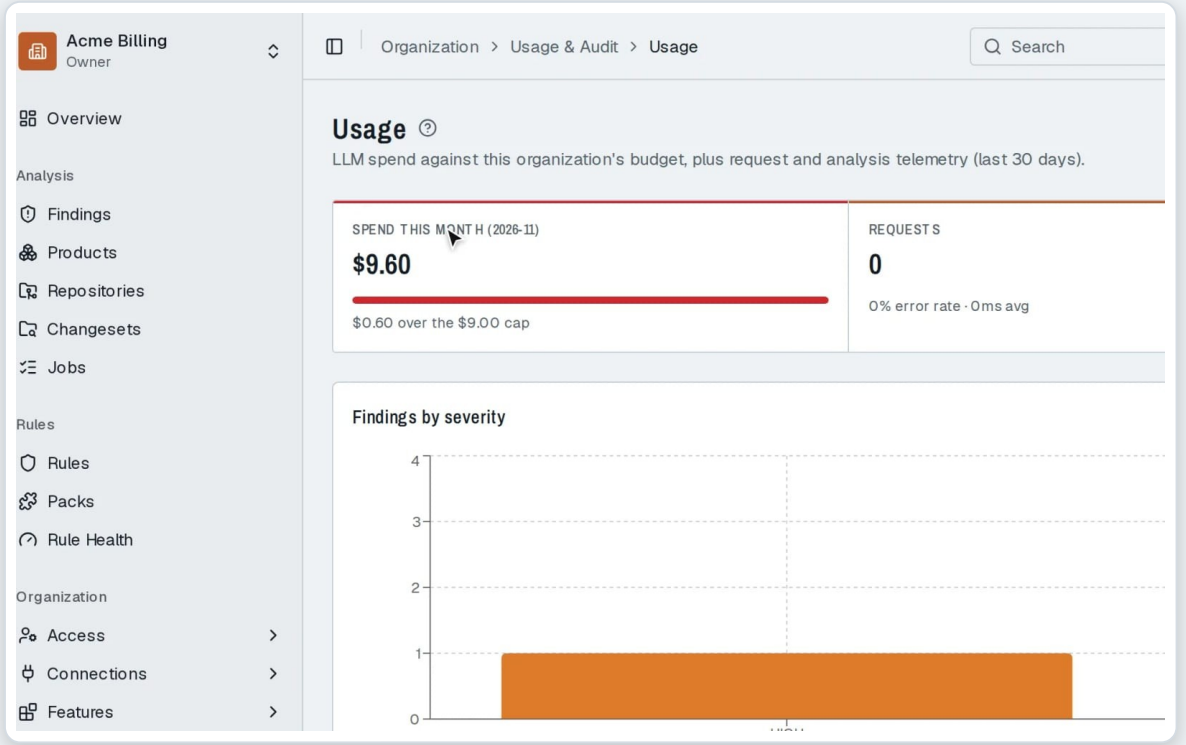
The report covers the whole change. Anyone with the link can read it until the link expires.

04 · FINANCE LEAD

Know what AI review costs, and cap it

“As a finance lead, I want to cap what AI review costs each month and choose where the model runs, so the bill holds no surprise.”

AI tools bill per use. Without a cap, a busy month or a runaway script shows up only on the invoice.

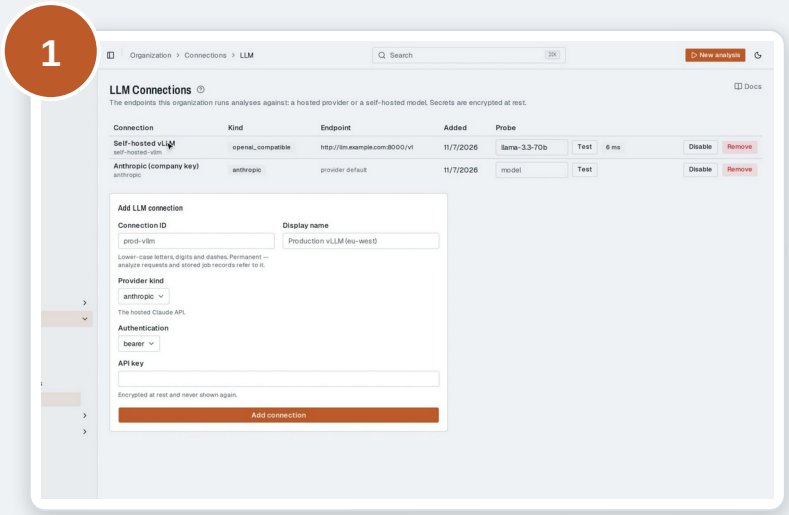


What to notice

- A hosted provider bills you directly, on your own key. A self-hosted model keeps the code on your network.
- The cap counts every model call, at the price the platform admin sets for each model.
- The cap is checked before a review starts: one already running finishes, so a month can end a little over.

Know what AI review costs, and cap it

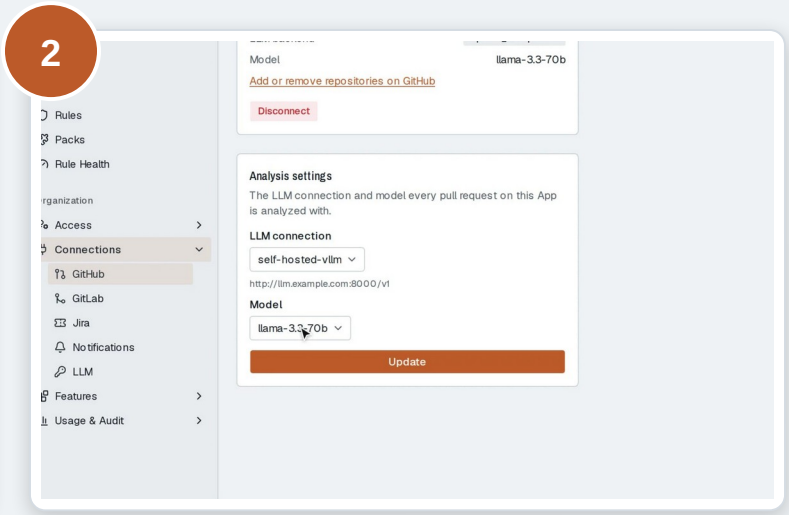
1



Choose where the model runs: a provider with your own key, or a model on your own network.

Two connections: a hosted provider on the company's own key, and a model on the company's network. A test call checks that it answers.

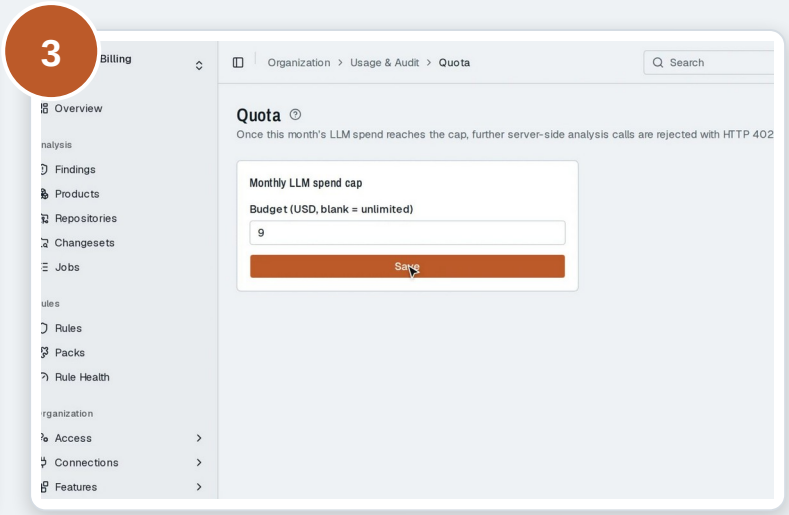
2



Reviews use the connection and the model you pick.

The GitHub settings say which connection and which model review the pull requests.

3

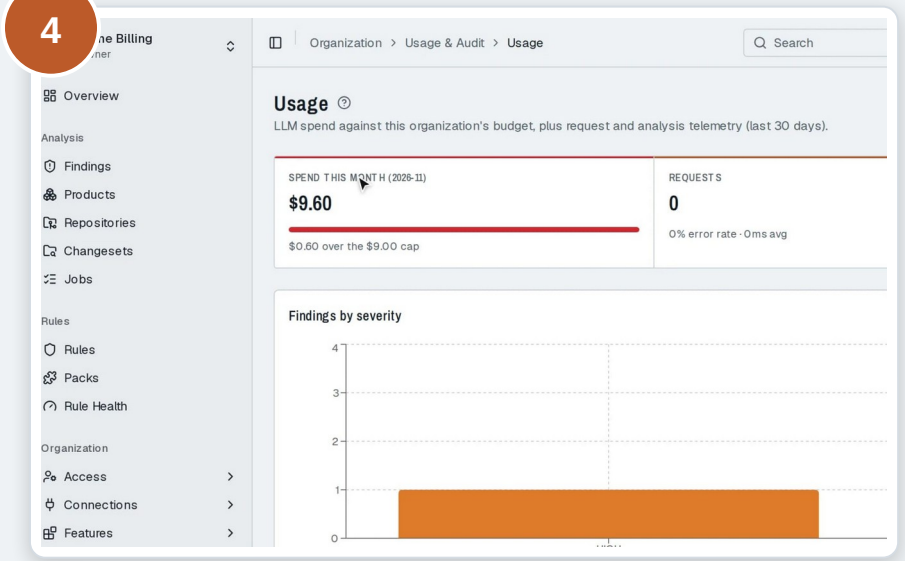


Finance sets a monthly cap on what the model may cost.

The finance lead sets a monthly cap in dollars, for every model call the organization makes.

Know what AI review costs, and cap it

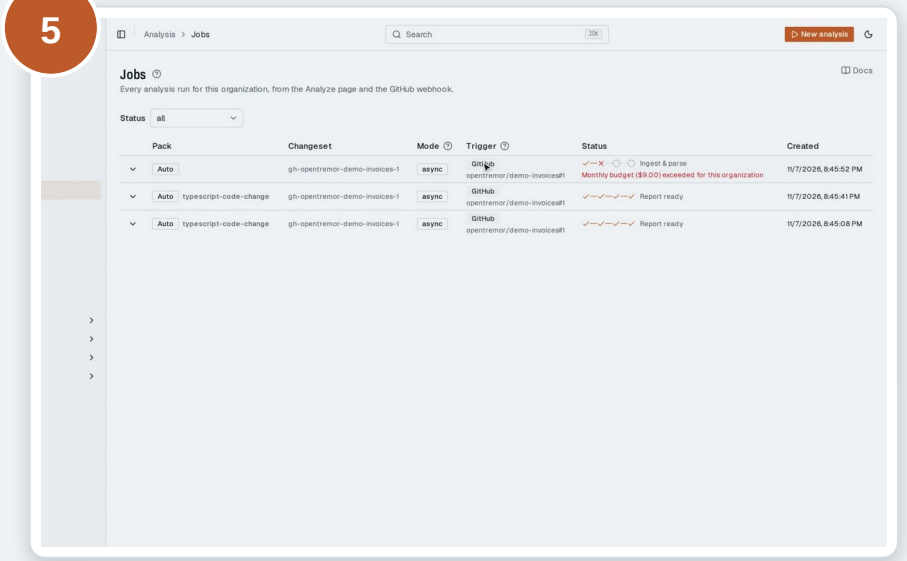
4



This month's spend, against the cap.

This month's spend against the cap. The last review started under the cap and finished just over it.

5



Once the cap is reached, new reviews stop until next month or a higher cap.

The next review does not start, and the job says why. Reviews run again next month, or once the cap is raised.

NEXT

Read more

Enterprise

A flat monthly price, onboarding and priority support.

opentremor.dev/enterprise

Dedicated instance

A single-tenant instance, operated by us in a supported cloud and region.

opentremor.dev/dedicated-instance

Self-hosted

Docker on one VM, or Helm on Kubernetes.

docs.opentremor.dev/deployment/

Pricing

Plans, and what each one includes.

opentremor.dev/pricing

See it in motion



opentremor.dev/tour/for/it-director · docs.opentremor.dev