QA / test lead



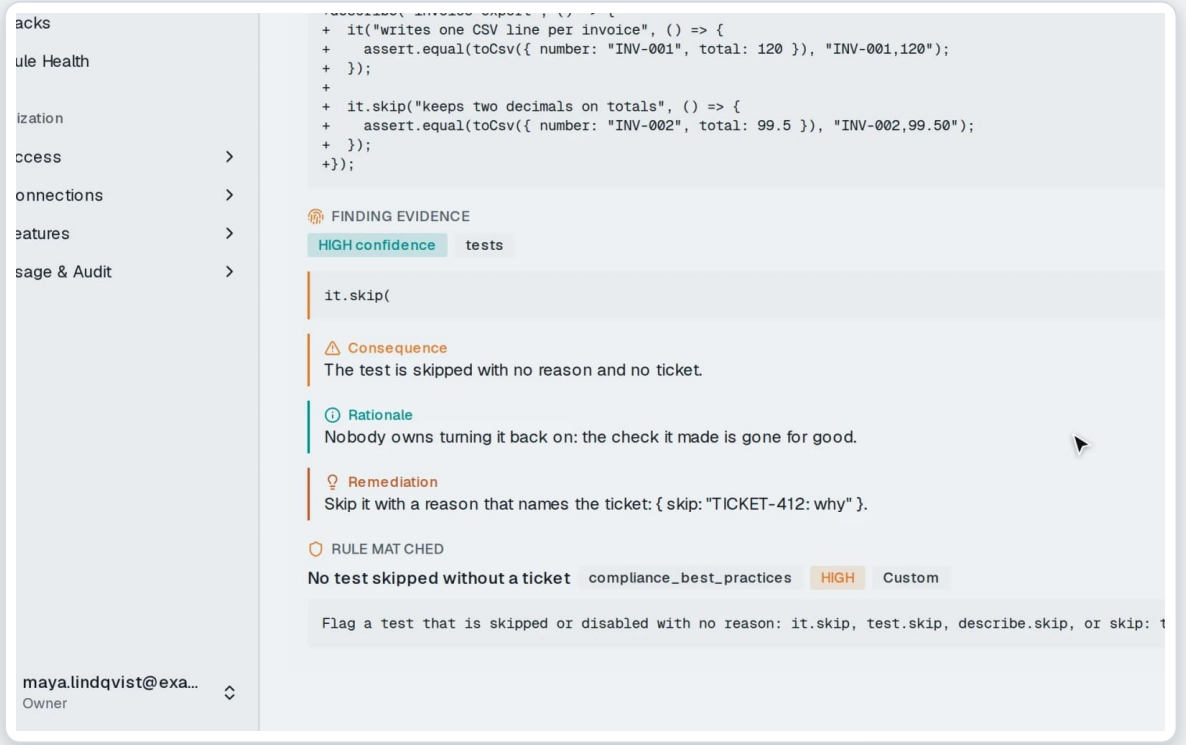
Your test rules checked on every pull request, and a merge check that holds.

01 · QA LEAD

A rule of your own: no test skipped without a ticket

“As a QA lead, I want a pull request that skips a test held unless it names a ticket, so coverage does not erode one test at a time.”

A skipped test passes CI. Nobody notices it in review, and nobody turns it back on.

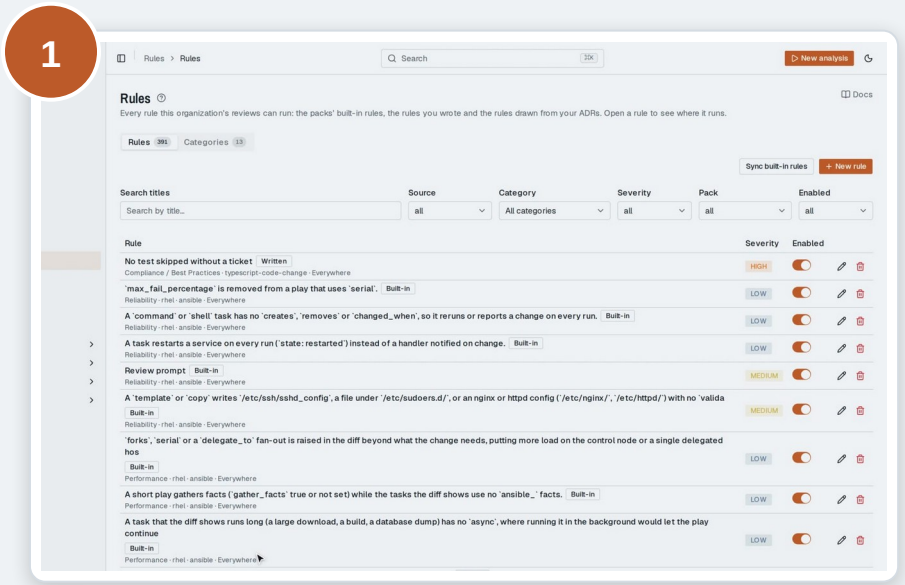


What to notice

- Anyone with the right role can write a rule. The next review checks it, with no code to write.
- A rule applies to the packs you pick: here, TypeScript files.
- HIGH and CRITICAL findings hold the merge. Lower ones only show in the review.

A rule of your own: no test skipped without a ticket

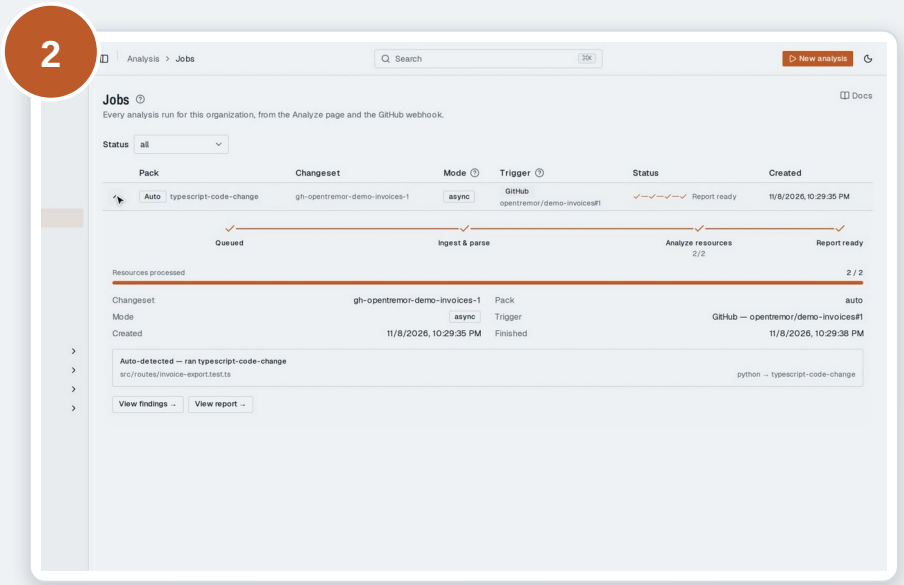
1



A QA lead writes a rule: no test is skipped without a ticket.

The QA lead writes the rule in plain words: what to flag, its severity, and the files it applies to.

2



The next pull request skips a test. OpenTremor reviews it against the rule.

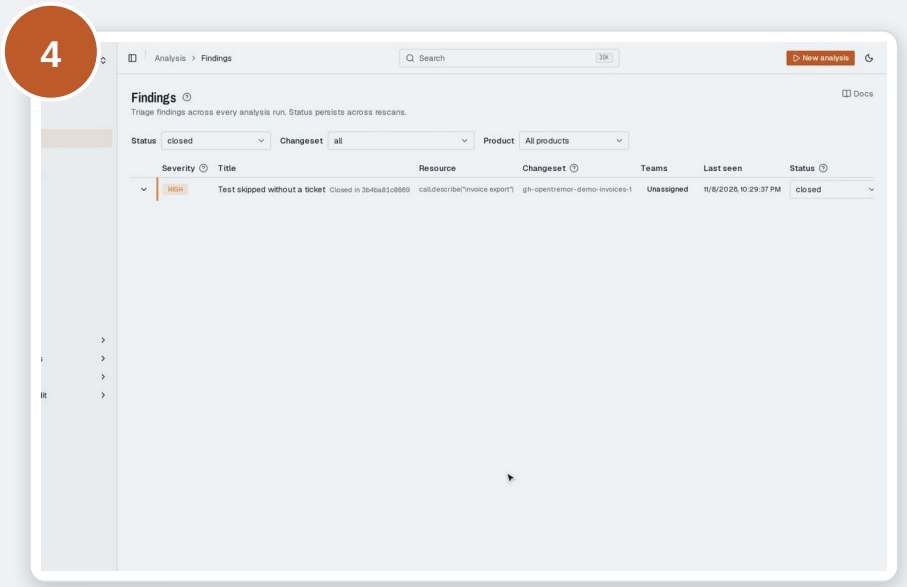
The next pull request adds tests and skips one of them. OpenTremor reviews it against the new rule.

A rule of your own: no test skipped without a ticket



The finding holds the merge, and says why the skip matters.

The finding quotes the skip and says why it matters. It is HIGH, so it holds the merge.



The next push names the ticket. The finding closes.

The next push gives the skip a reason that names the ticket. The finding closes and the merge check passes.

02 · REVIEWER

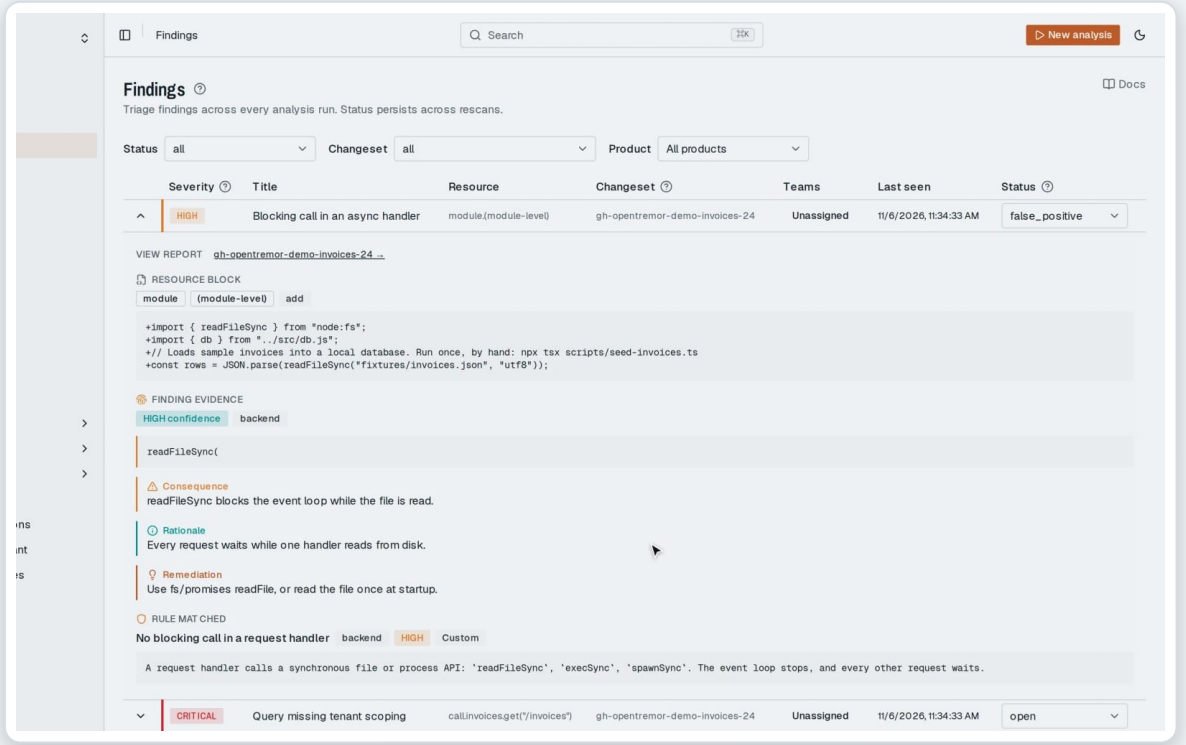
Dismiss a false positive once

“As a reviewer, I want to dismiss a false positive with a reason, so it does not come back after every push.”

Most review tools forget. The same false positive comes back on the next commit, and people start to ignore the tool.

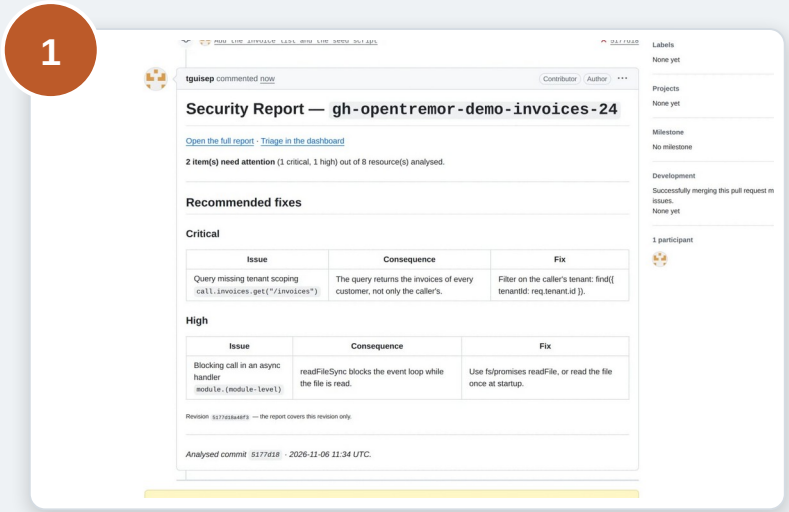
What to notice

- The note stays with the finding, for the next reviewer and for audits.
- Each decision counts against the rule. When reviewers often reject a rule, OpenTremor suggests a change to it.



Dismiss a false positive once

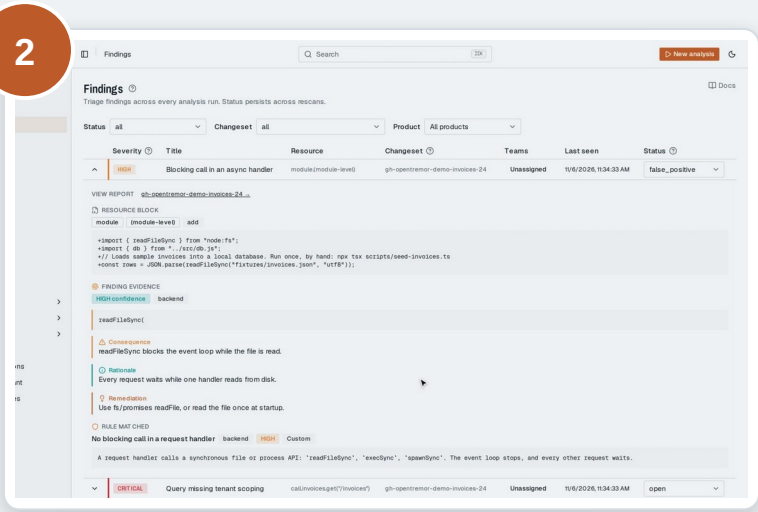
1



Every pull request is reviewed against your rules.

OpenTremor's comment on the pull request lists two findings: the customer data bug and a slow, blocking call in a script.

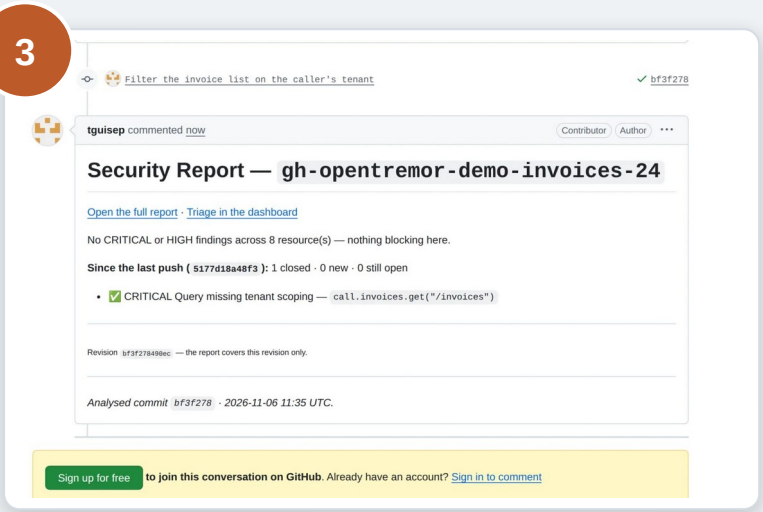
2



A reviewer dismisses a false positive once, with a reason.

The script runs only once, by hand, so the blocking call is not a problem. The reviewer marks it as a false positive and writes why.

3



The next push: the fix closes the finding, and the dismissal stays.

The developer pushes the fix for the customer data bug, and that finding closes. The dismissed finding stays dismissed, with its note.

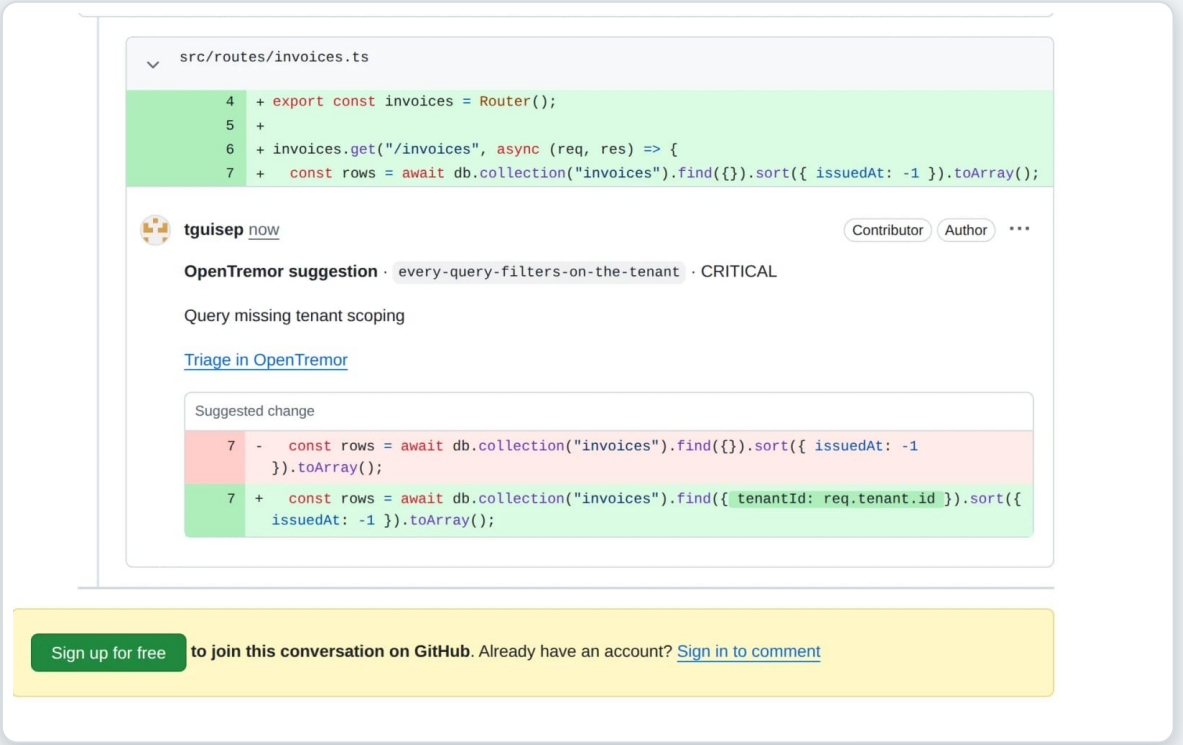
Every pull request checked against your team's rules

“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

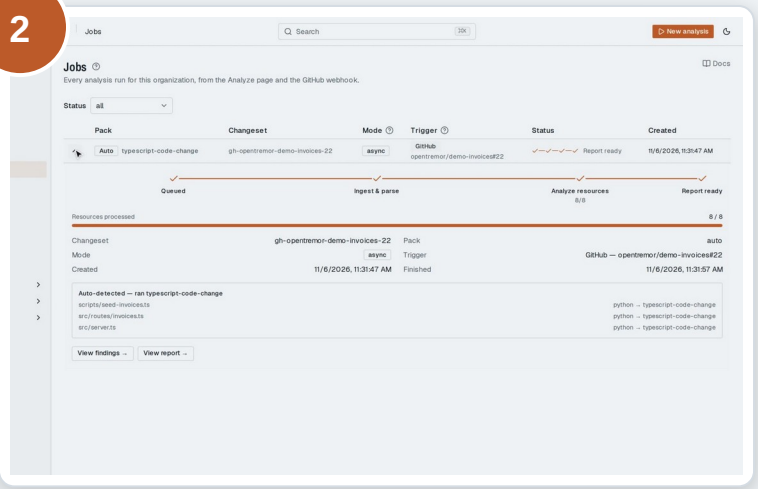
1



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.

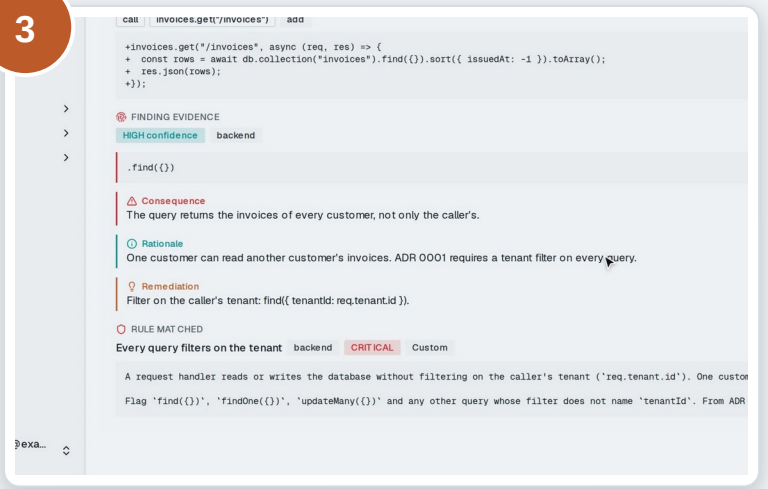
2



OpenTremor reviews it against your team's rules.

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

3

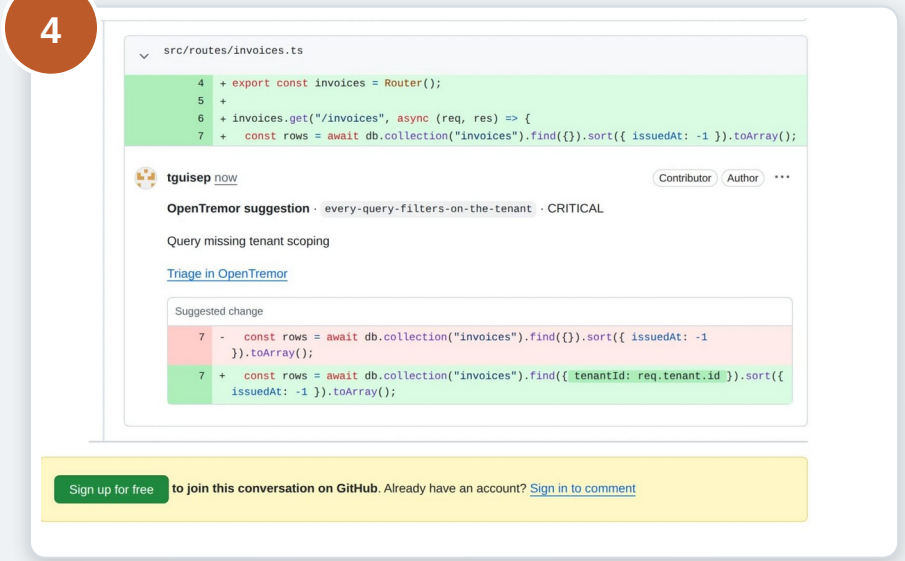


Each finding names the rule it breaks and why it matters.

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

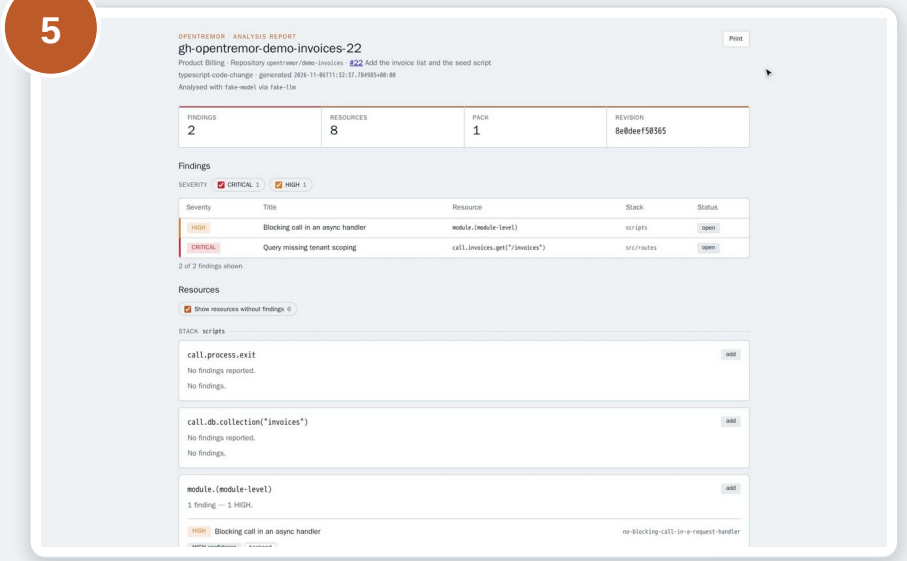
4



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.

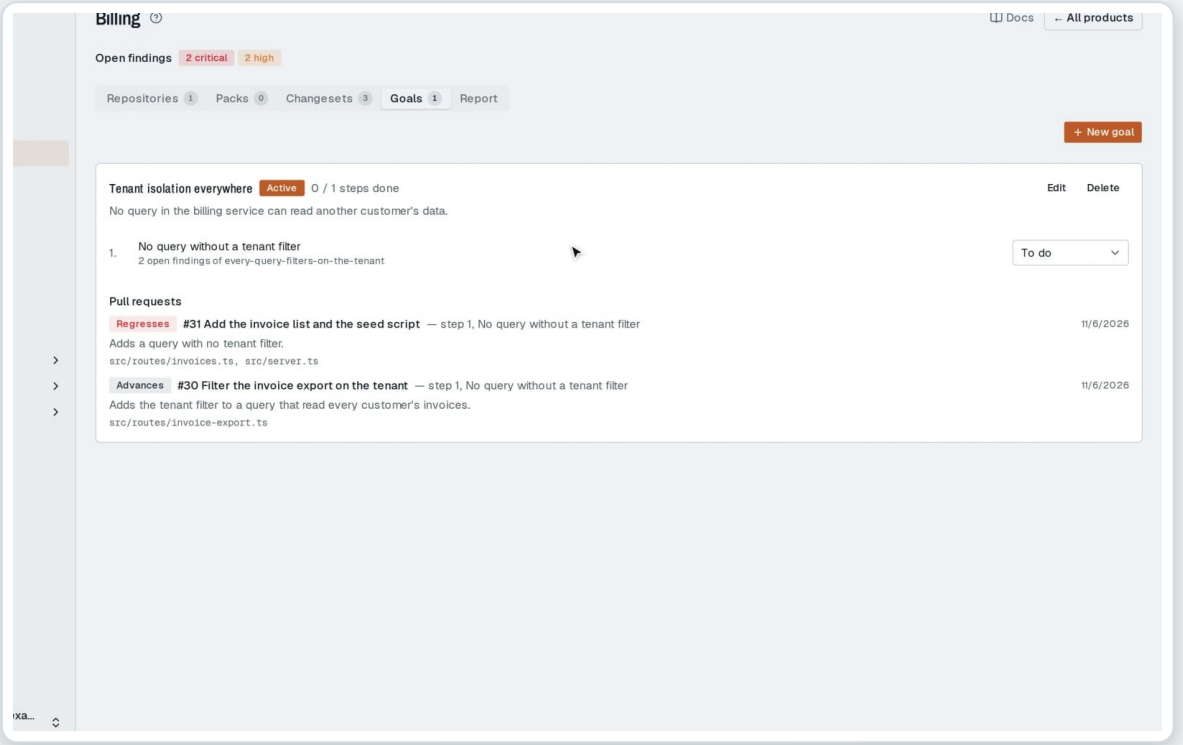
Track an engineering goal across pull requests

“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.

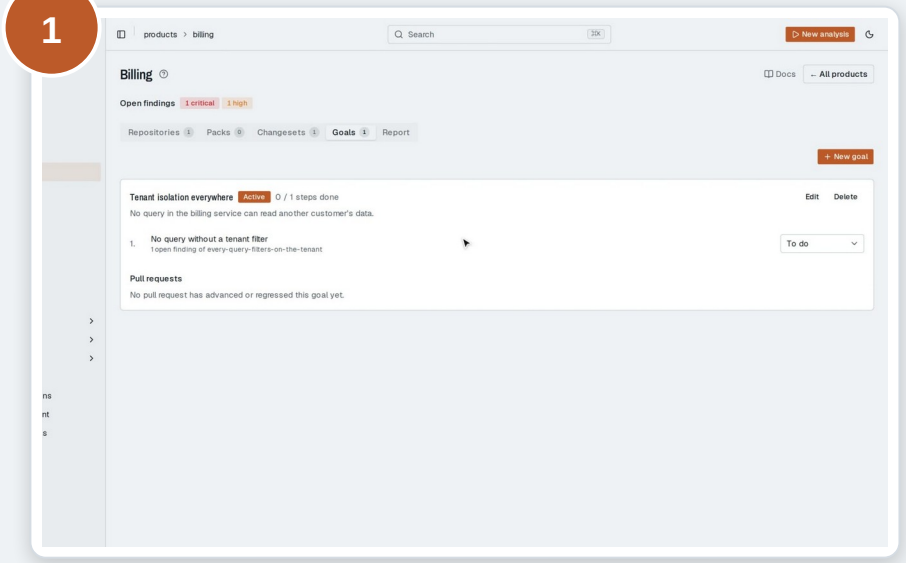
What to notice

- 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

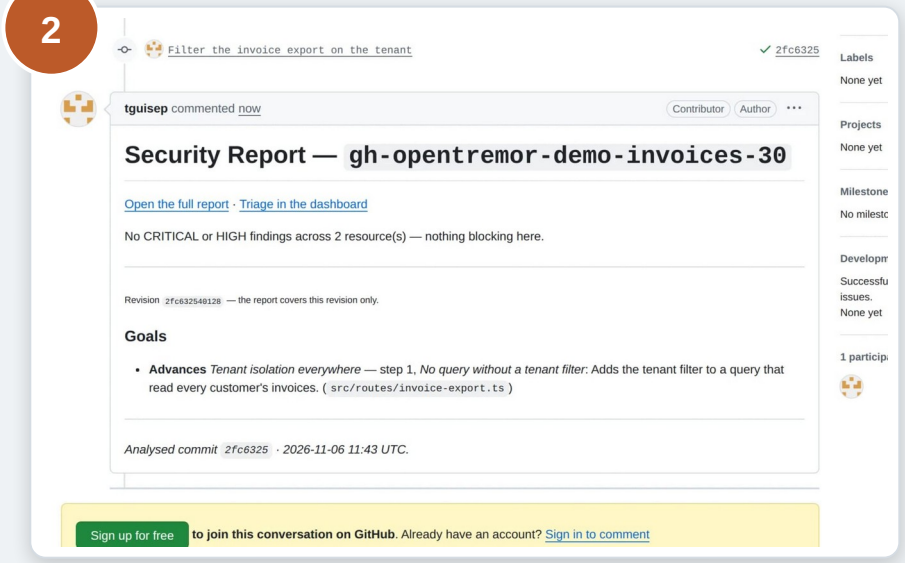
1



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

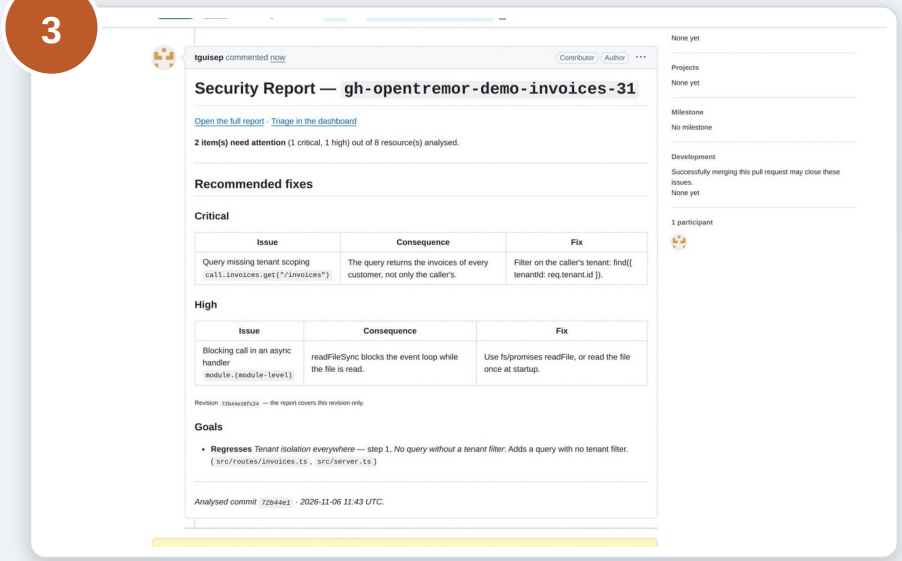


A pull request that fixes old code advances the goal.

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

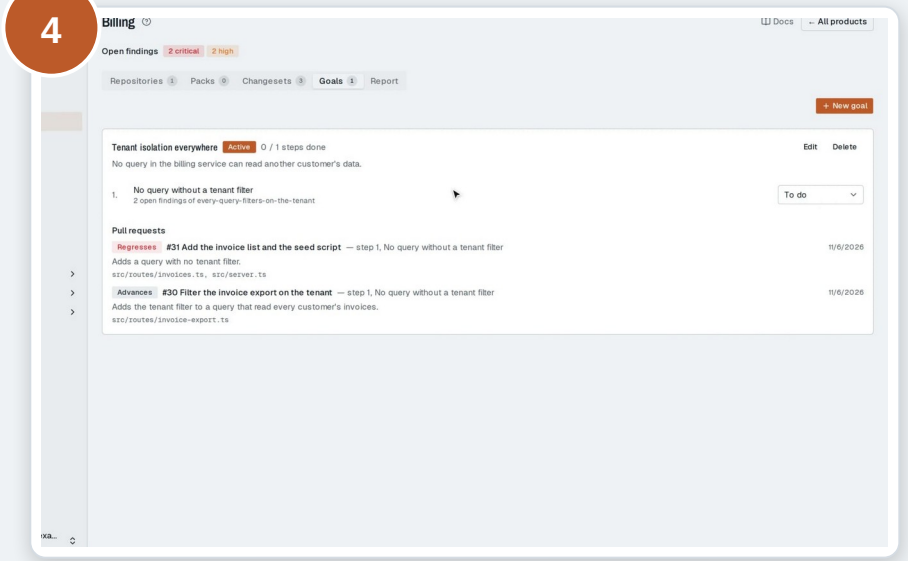
3



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.

NEXT

Read more

Write a rule

Step by step, with screenshots.

docs.opentremor.dev/guide/write-rule/

How rules apply

Which rules a review runs, and where each setting lives.

docs.opentremor.dev/configuration/rules/

Dismiss a false positive

Step by step, with screenshots.

docs.opentremor.dev/guide/triage/

See it in motion



opentremor.dev/tour/for/qa · docs.opentremor.dev