softup Self-Healing

User guide

How to set up Self-Healing for your team, and how to go from a bug to a reviewed fix every day after.

For the teams Softup invites · November 2026
selfheal.softup.co

BEFORE YOU START

What Self-Healing does

Self-Healing turns a bug into a fix your team only has to review. It collects bugs from your app, your GitHub issues and your logs, works on them with Claude Code in a sealed-off environment, runs your own tests, and hands you the result as a **draft pull request**. Nothing is merged for you: every change waits until someone on your team says yes.

1. A bug comes in
From the report button in your app, a GitHub issue, or an error in your logs.

2. It becomes a ticket
A request in Self-Healing and an issue in your repository.

3. A run fixes it
Claude Code works on the issue and proposes a change.

4. Your tests check it
A second run validates the change with your own tests.

5. You decide
A draft pull request waits for your review.

This guide has two parts. **Part 1** is for the person who sets Self-Healing up for the team, usually the tech lead who received the invitation: it is done once. **Part 2** is for everyone on the team, every day.

Contents

PART 1 · SETTING UP

1 Accept your invitation
2 Setting up
3 Connect GitHub
4 Add your Claude key
5 Create a project
6 Let Self-Healing learn your project
7 Your first fix
8 Invite your team
9 Bring in bugs
10 Fix new bugs automatically

PART 2 · EVERYDAY USE

11 Triage: from error to ticket
12 Start a fix
13 Follow a run and read its log
14 Review the draft pull request
15 Runs, audit log, emails and limits

REFERENCE

16 Request statuses
17 Roles
18 When something is refused

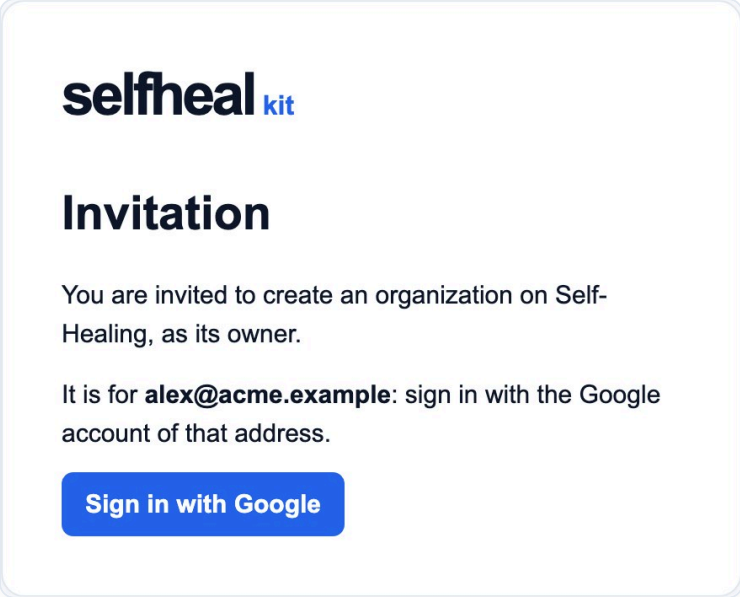
PART 1 · SETTING UP

1. Accept your invitation

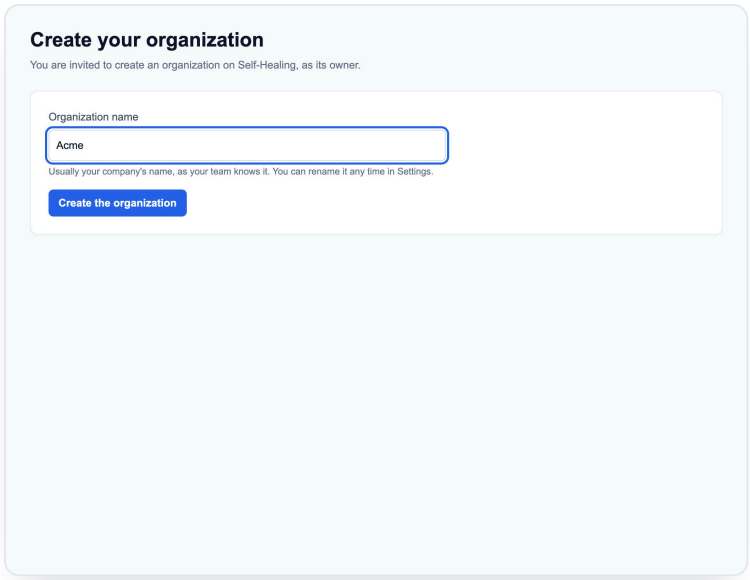
Who: the person who received the invitation

Self-Healing is invite-only. Softup emails you a personal invitation that works for the Google account of your email address. It also lists what to have ready: access to the GitHub repository you want to use (or knowing who owns it), an Anthropic API key, and about 15 minutes.

1. Click **Accept the invitation** in the email (or open the link you were sent) and click **Sign in with Google**. Use the Google account of the address the invitation names.
2. On **Create your organization**, check the **Organization name**: it is suggested from your email's domain, and you can rename it any time in **Settings**.
3. Accept the Terms of Service and the privacy notice, if asked, and click **Create the organization**. You are now its **owner**, and the guided setup opens.



The invitation page.



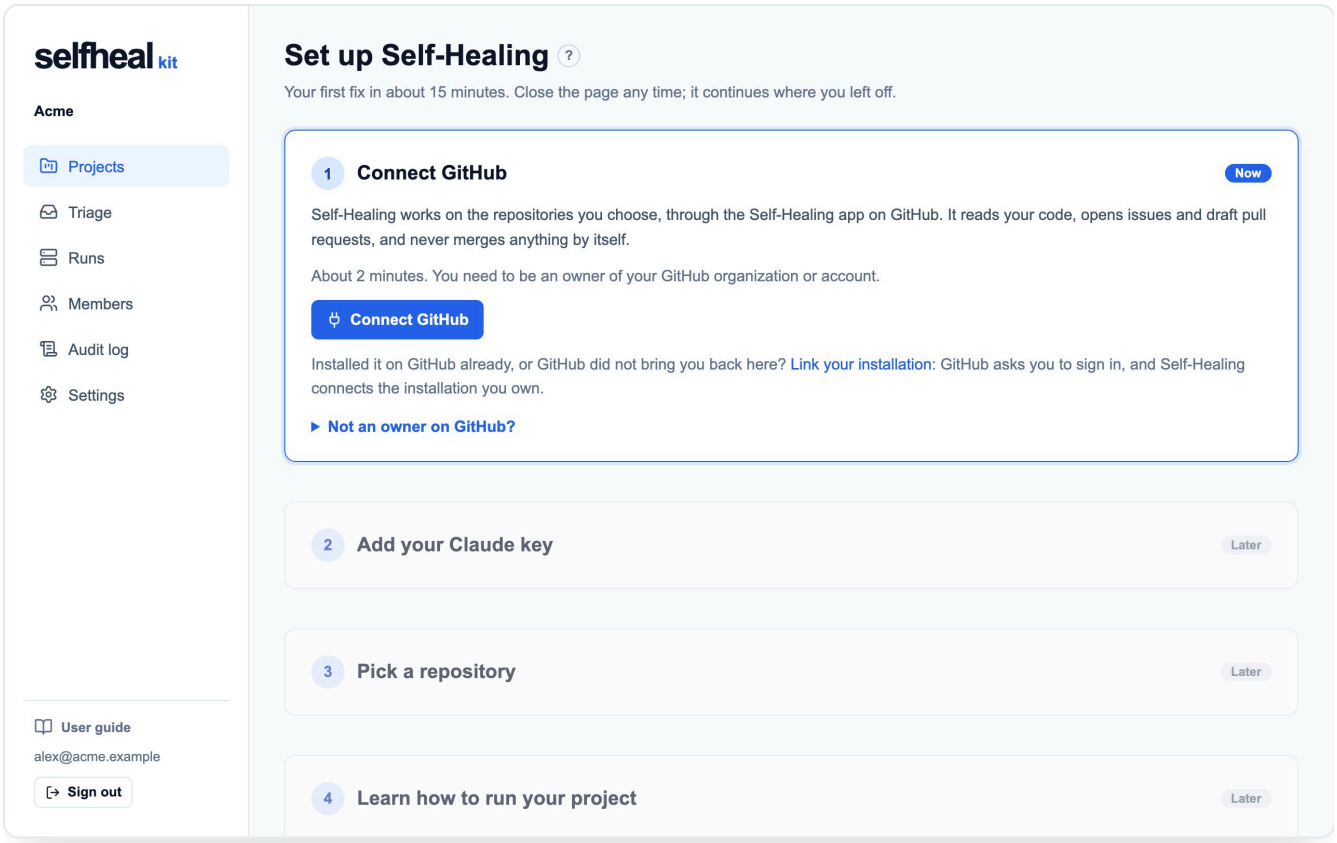
Creating the organization: only its name.

> **Good to know:** an invitation works for 7 days and only once. If it has expired or was used, ask Softup for a new one.

2. Setting up

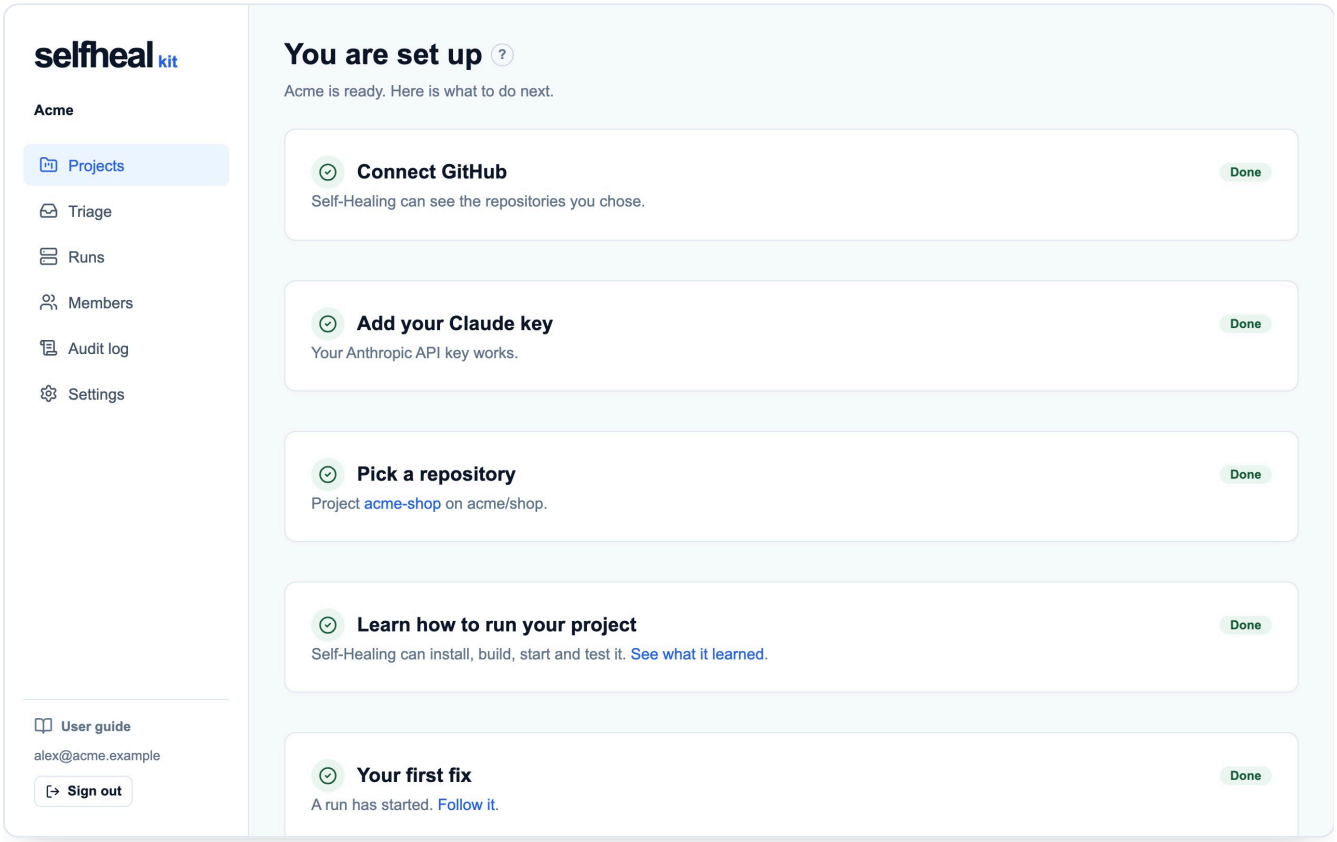
Who: an owner or admin

The setup page takes you from an empty organization to your first fix in five steps, each with what it is for and how long it takes. The step to do now is open; the others wait their turn, and a step done shows a tick. You can close the page at any time: the **Getting started** list on the Projects page brings you back where you left off, with **Continue the setup**.



The guided setup: five steps to a first fix.

Each step is described in the next chapters. Once they are all done, the page says **You are set up** and suggests what to do next: the report button, your logs and your team.



The setup done: every step ticked.

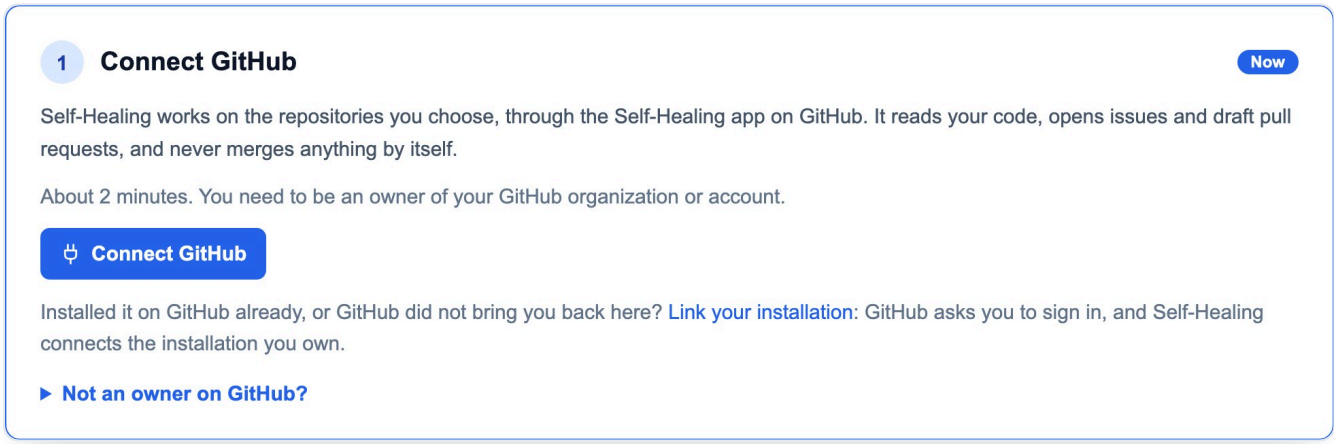
> **Prefer to do it together?** click **Talk to us** at the bottom of the setup page, and we set it up with you on a call.

3. Connect GitHub

Who: an owner or admin who is an owner on GitHub

Self-Healing works on the repositories you choose, through the Self-Healing app on GitHub. It reads your code, opens issues and draft pull requests, and never merges anything by itself.

1. In step 1 of the setup, click **Connect GitHub**.
2. GitHub asks you to install the app: choose your GitHub organization or account and **only the repositories** Self-Healing may work on.
3. GitHub asks you to authorize the app, then brings you back to the setup, which says **Connected the GitHub installation of ...**.



Step 1: connecting GitHub.

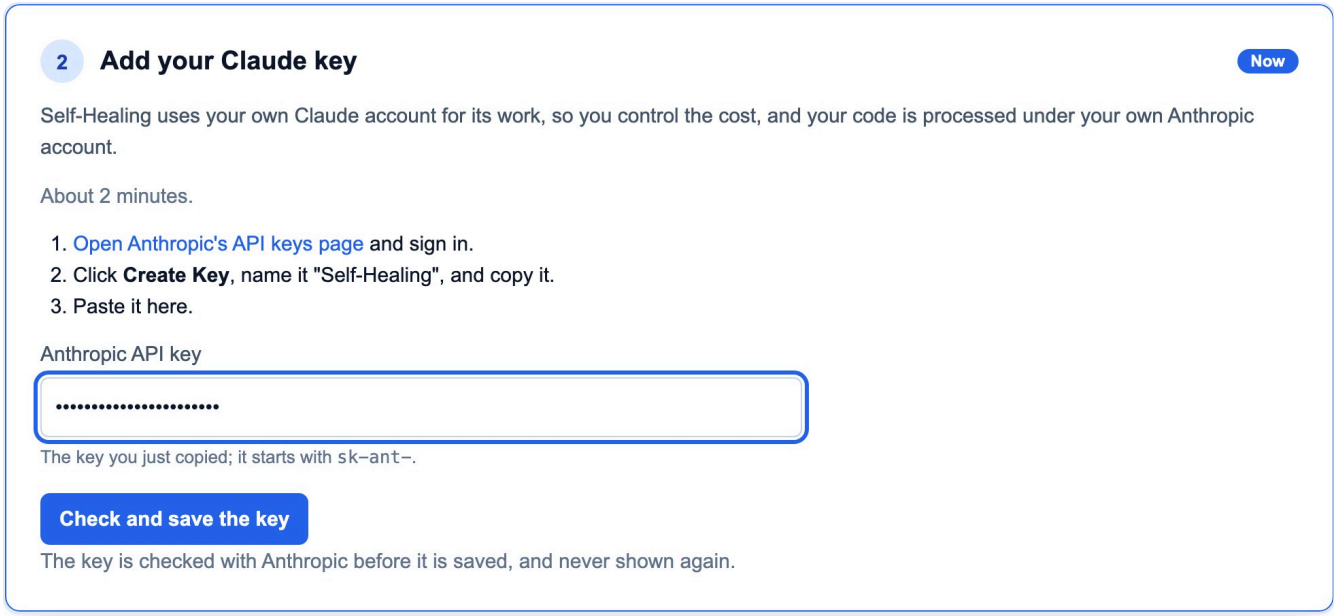
> **Not an owner on GitHub?** only an owner of the GitHub organization can install the app. Open **Not an owner on GitHub?** in step 1: it has a ready message to send them, and a button to invite them to your organization as an admin, so they can connect it themselves. Later, **Settings** shows the installation and lets you connect another one.

4. Add your Claude key

Who: an owner or admin

Self-Healing uses **your own** Anthropic API key for every Claude call your projects make: analysing errors, fixing, validating and reviewing. It is billed to you by Anthropic directly, and never used for another organization.

1. In step 2, click **Open Anthropic's API keys page** and sign in.
2. Click **Create Key**, name it "Self-Healing", and copy it.
3. Paste it into **Anthropic API key** and click **Check and save the key**. The key is checked with Anthropic before it is saved: **Key works** means it is ready.



Step 2: adding your Anthropic API key.

> **Good to know:** the key is encrypted and never shown again. In **Settings**, **Manage Claude connections** lists your keys by their last four characters; you can add several, which are tried in order, a rate-limited one being skipped until it resets. Without a key, fix runs are refused and error analyses wait.

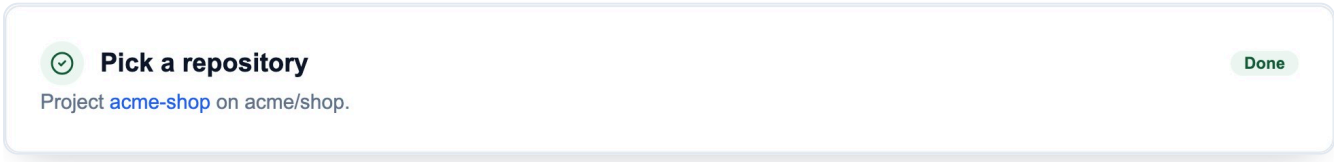
5. Create a project

Who: an owner or admin

A project is one repository: its bugs and their fixes. Creating one is a single choice.

If you gave Self-Healing **one** repository on GitHub, there is nothing to choose: the project is created as soon as GitHub is connected, and step 3 is done for you. With several:

1. In step 3, choose the **Repository**. Only the repositories you gave Self-Healing on GitHub are offered.
2. Click **Create the project**. It is named after the repository, and learning it starts at once.



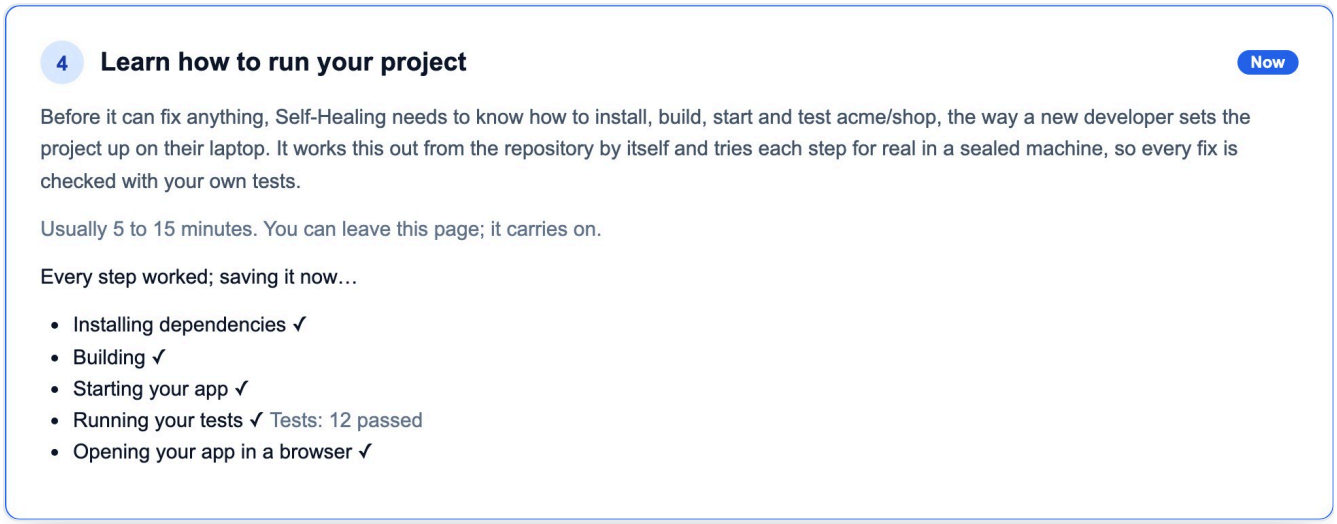
Step 3, done by itself: the project was made from the one repository.

More projects are added the same way, from **New project** on the Projects page. The keys a project needs for the report button are shown where you add the button (chapter 9).

6. Let Self-Healing learn your project

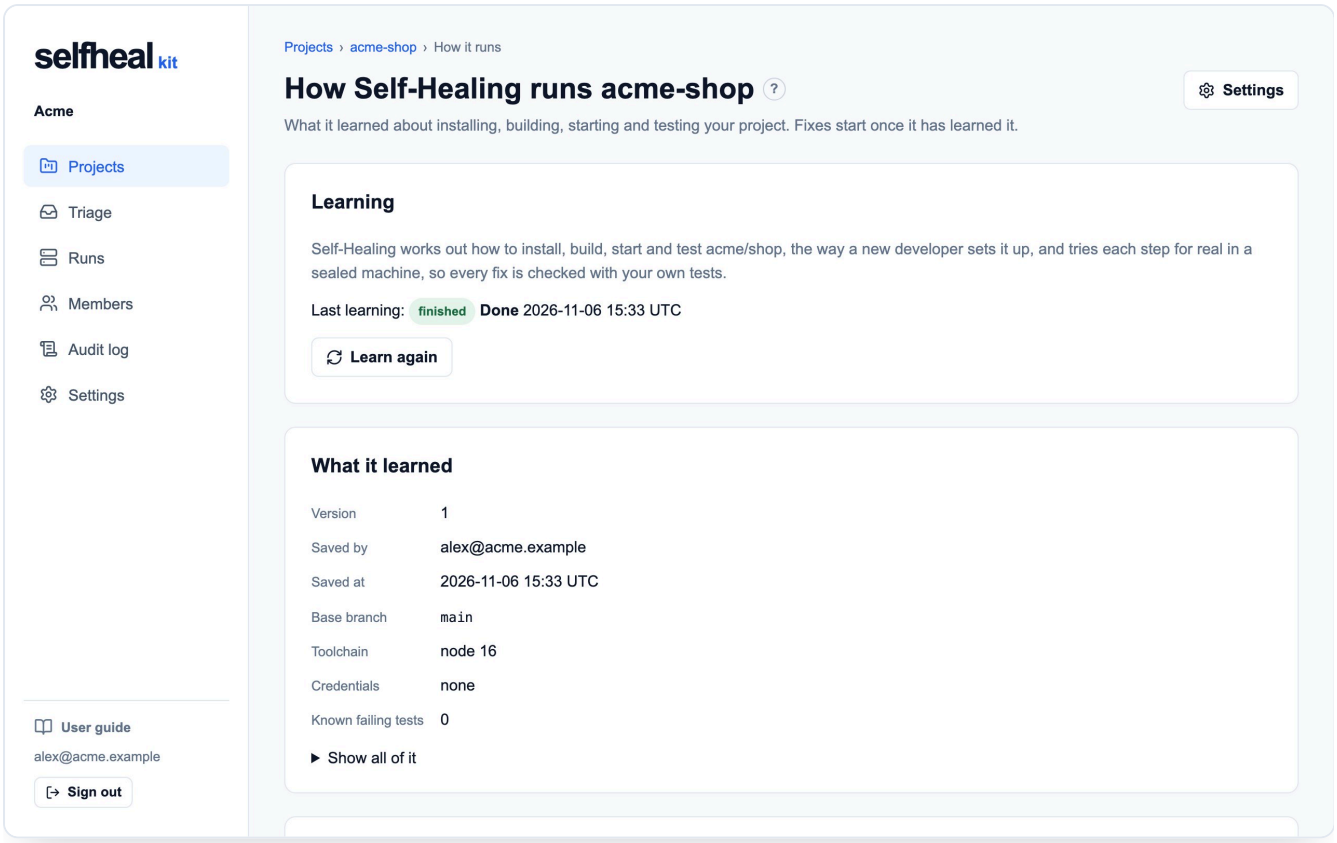
Who: an owner or admin

Before Self-Healing can fix anything, it needs to know how to install, build, start and test your product, so that every fix is checked with your own tests. It works this out by itself: in step 4, a short run reads your repository and tries each step for real. It usually takes 5 to 15 minutes, and the page shows its progress in plain words.



Step 4: Self-Healing learning the project.

When every step worked, what it learned is saved as the project's **environment profile**, and the setup moves on. When something needs a look, the step says what, with a **Review the setup** button: open it, check the steps and the commands, adjust them if needed, and click **Save as a new version**.



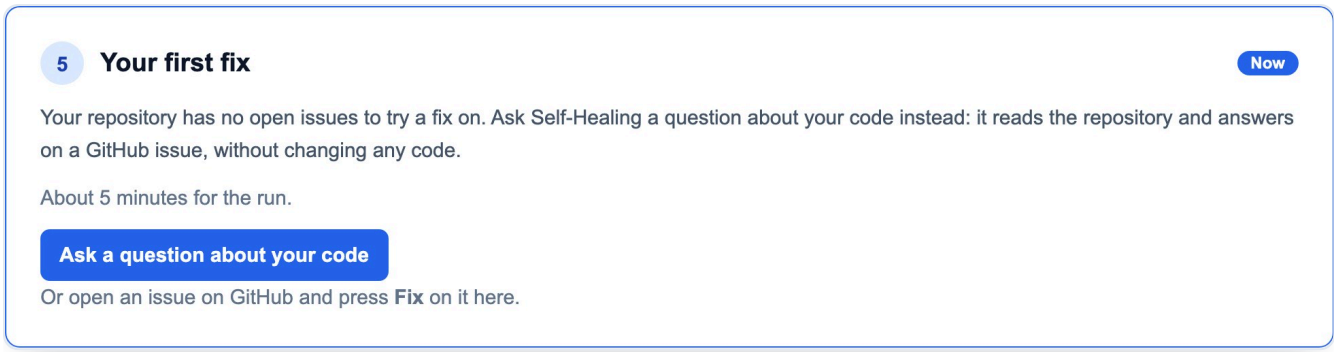
The project's How it runs page: what Self-Healing learned.

> **Good to know:** the project's **How it runs** page shows what it learned; developers can change it by hand, under **Edit by hand**, and add notes for the agent, and files like `CLAUDE.md` and `AGENTS.md` in your repository are read as well. Credentials are listed by name only. **Learn again** works it out anew after a big change. The profile also sets the fix limits: by default 3 attempts per issue and 50 runs a month.

7. Your first fix

Who: an owner or admin

Step 5 starts your first run, so you see the whole loop once. If your repository has open issues, it lists a few: pick a small one and click **Fix this**. Self-Healing writes the fix, runs your tests on it and opens a draft pull request for you to review. With no open issue, click **Ask a question about your code**: it reads the repository and answers on a GitHub issue, without changing any code.



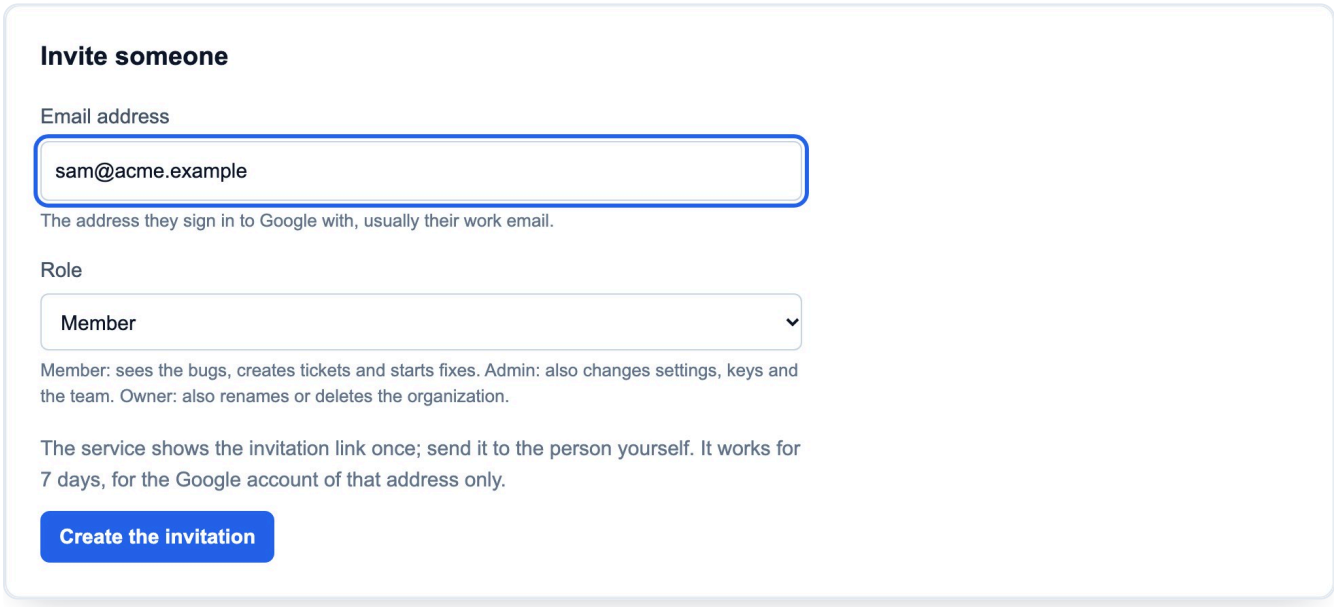
Step 5: your first fix.

Chapter 13 explains how to follow the run.

8. Invite your team

Who: an owner or admin

1. Open **Members** in the menu.
2. Under **Invite someone**, enter their **Email address**, choose a **Role** (see chapter 17), and click **Create the invitation**.
3. We email them the invitation. Its link is also shown **once**, to send yourself should the email not arrive. It works for 7 days, for the Google account of that address.



Inviting a teammate.

Pending invitations are listed with a **Revoke** button. Admins can change a member's role or remove them; the last owner cannot leave until another member is an owner.

9. Bring in bugs

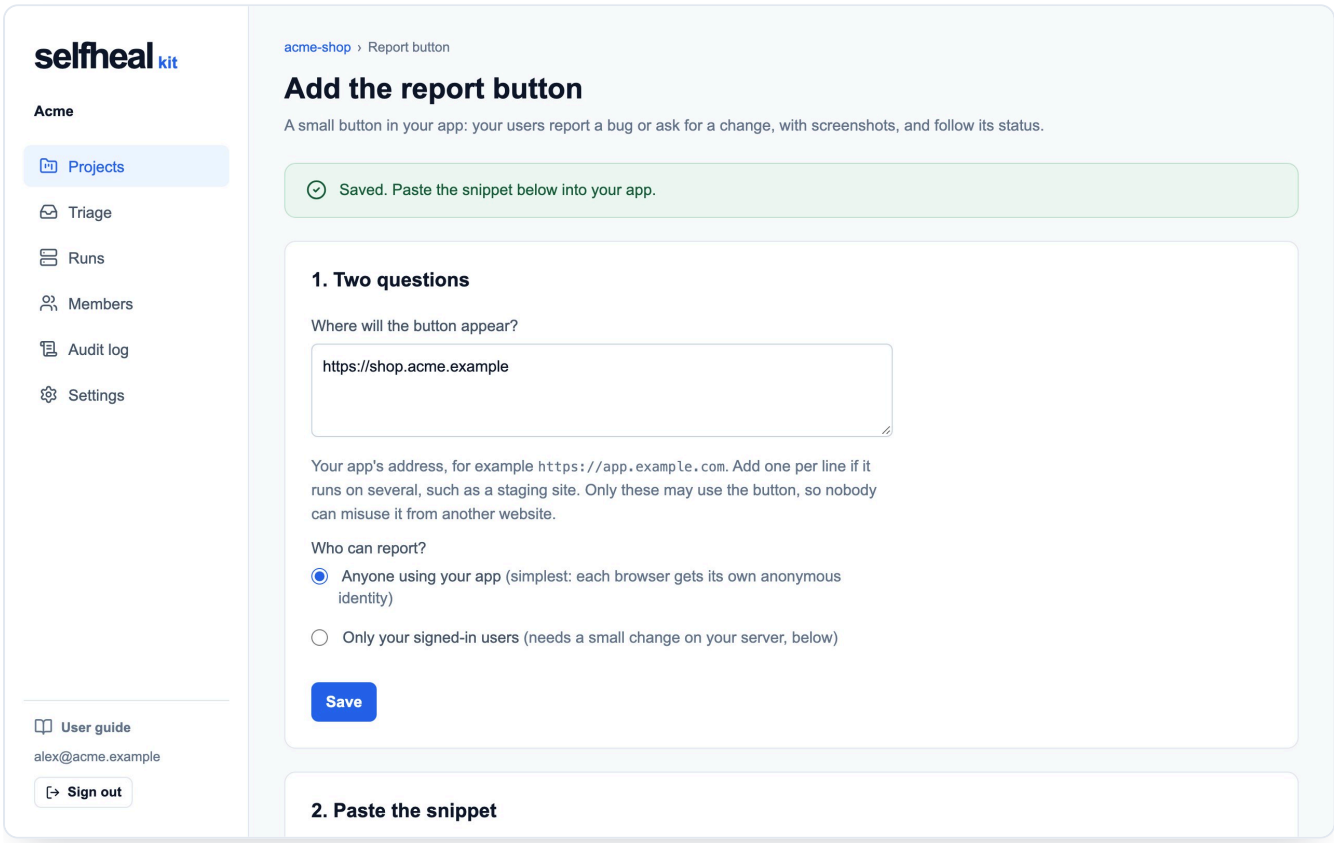
Who: an owner or admin

Bugs can reach Self-Healing in three ways. Use any or all of them.

The report button in your app

Your users press it to report a bug or ask for a change, with screenshots, and can follow its status.

1. Open the project and click **Add the report button**.
2. Answer **Where will the button appear?** with your app's address, such as `https://shop.example.com`, one per line if it runs on several. Only these may use the button.
3. Answer **Who can report?**: **Anyone using your app** (simplest), or **Only your signed-in users**, which needs a small change on your server, explained on the page.
4. Click **Save**, and paste the snippet the page shows into your app's pages.



Adding the report button: two questions and a snippet to paste.

> **Good to know:** if the button is used from an address you did not list, the project's page offers **Allow this address**, so a new site can be allowed in one click.

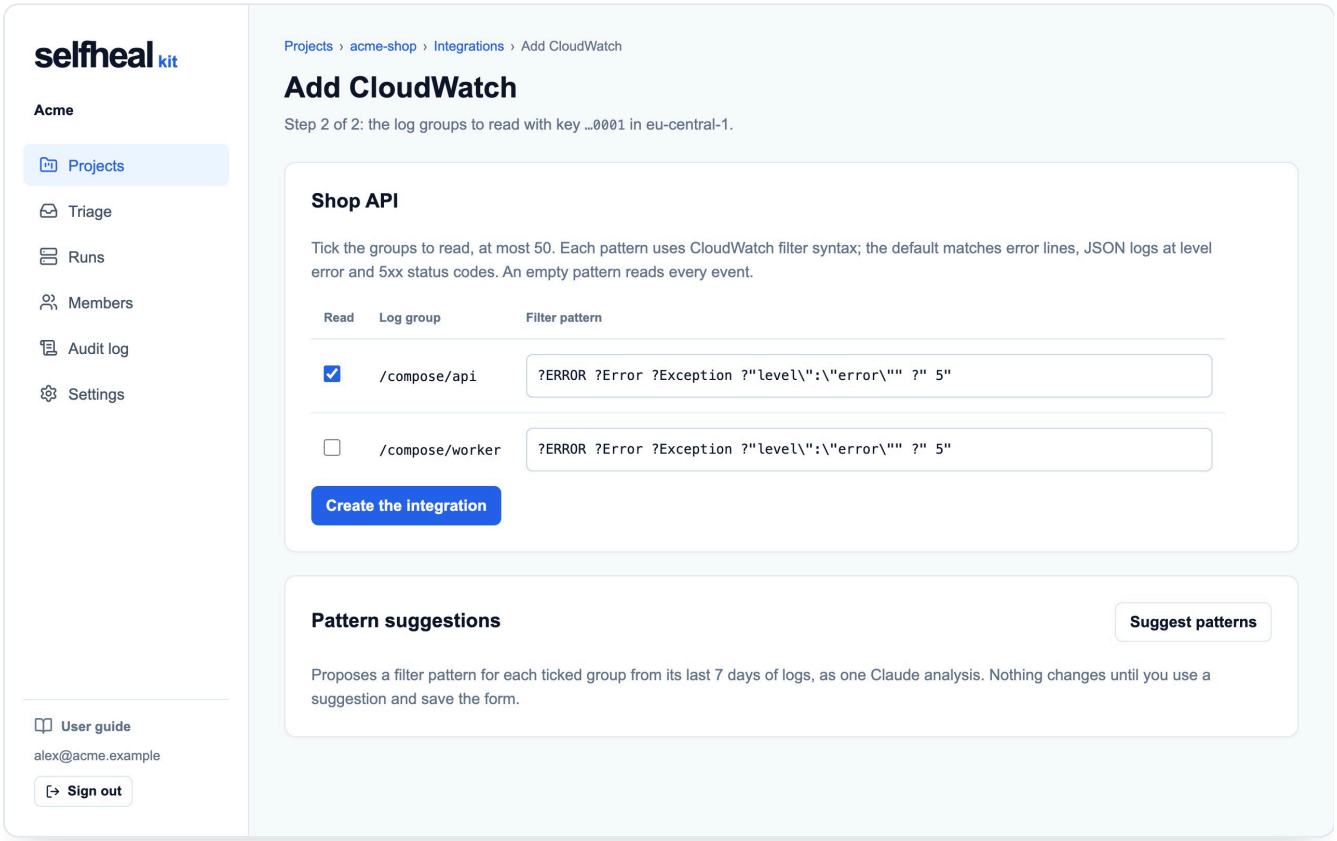
GitHub issues

Issues opened in the repository are tracked automatically. When the project fixes new bugs automatically (chapter 10), an issue opened by an owner, member or collaborator of the repository starts a fix by itself, unless it has the `selfheal-skip` label.

Errors in your AWS CloudWatch logs

When **Add CloudWatch** offers **Connect AWS (recommended)**, use it: one click opens AWS with a ready-made template that creates a read-only role for Self-Healing, with no key to create or rotate. Create the stack, copy its **RoleArn** from the stack's Outputs into the form, choose the **Region** of your logs and click **List log groups**; then continue from step 3 below. Otherwise, use an access key:

1. In AWS, create an IAM user for Self-Healing only, with an access key.
2. In the project, open **Integrations** and click **Add CloudWatch**. Enter a **Label**, the **Access key ID**, the **Secret access key** and the **Region**, then click **List log groups**.
3. The log groups that look like your app's are ticked for you; change the choice if needed. The default filter pattern matches error lines, JSON logs at level error and 5xx status codes; **Suggest patterns** proposes one from your own logs. Click **Create the integration**.
4. The connection is tested at once. If the page asks for it, attach the **IAM policy** shown to the IAM user, then click **Test connection**.



Choosing the log groups to read.

> **Good to know:** personal data such as email and IP addresses is removed from log lines before they are stored or analysed. You can add your own patterns to remove in the project's **Settings**, under **Advanced settings**, where you can also set a Slack webhook for alerts and the daily analysis budget.

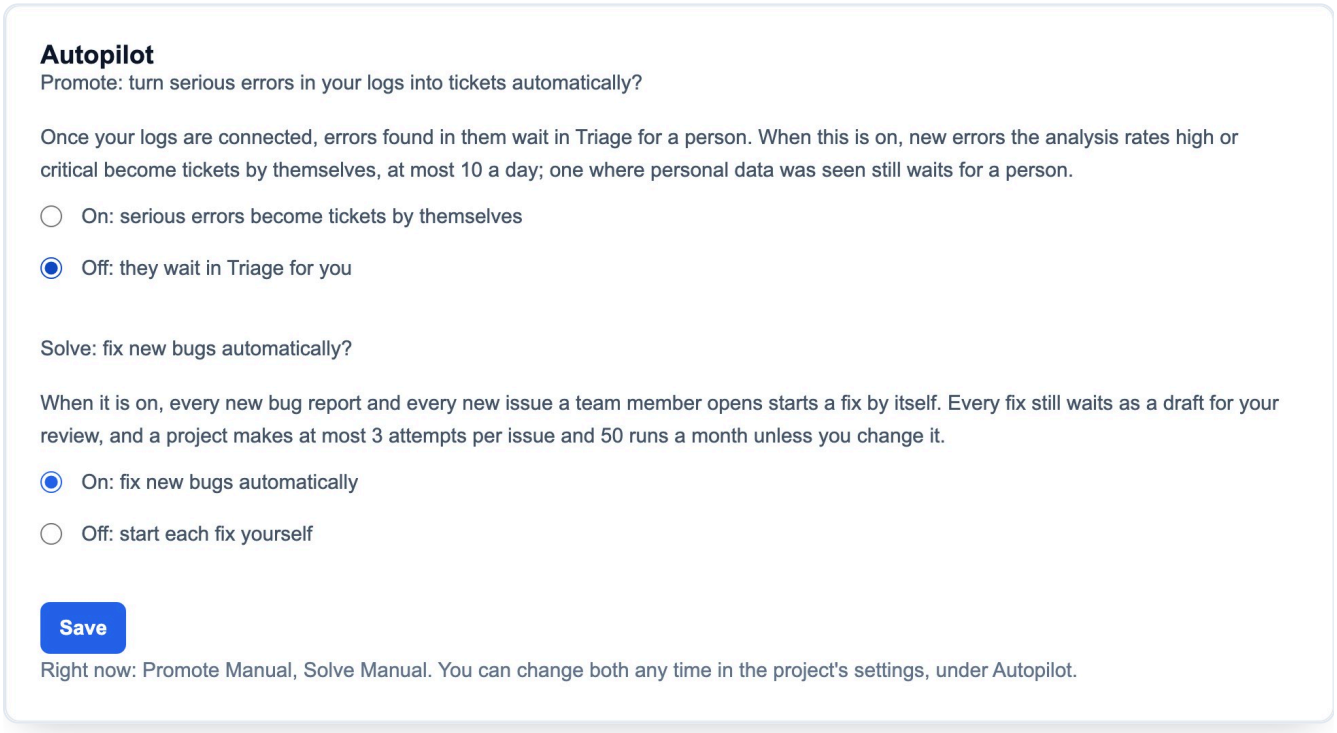
10. Fix new bugs automatically

Who: an owner or admin

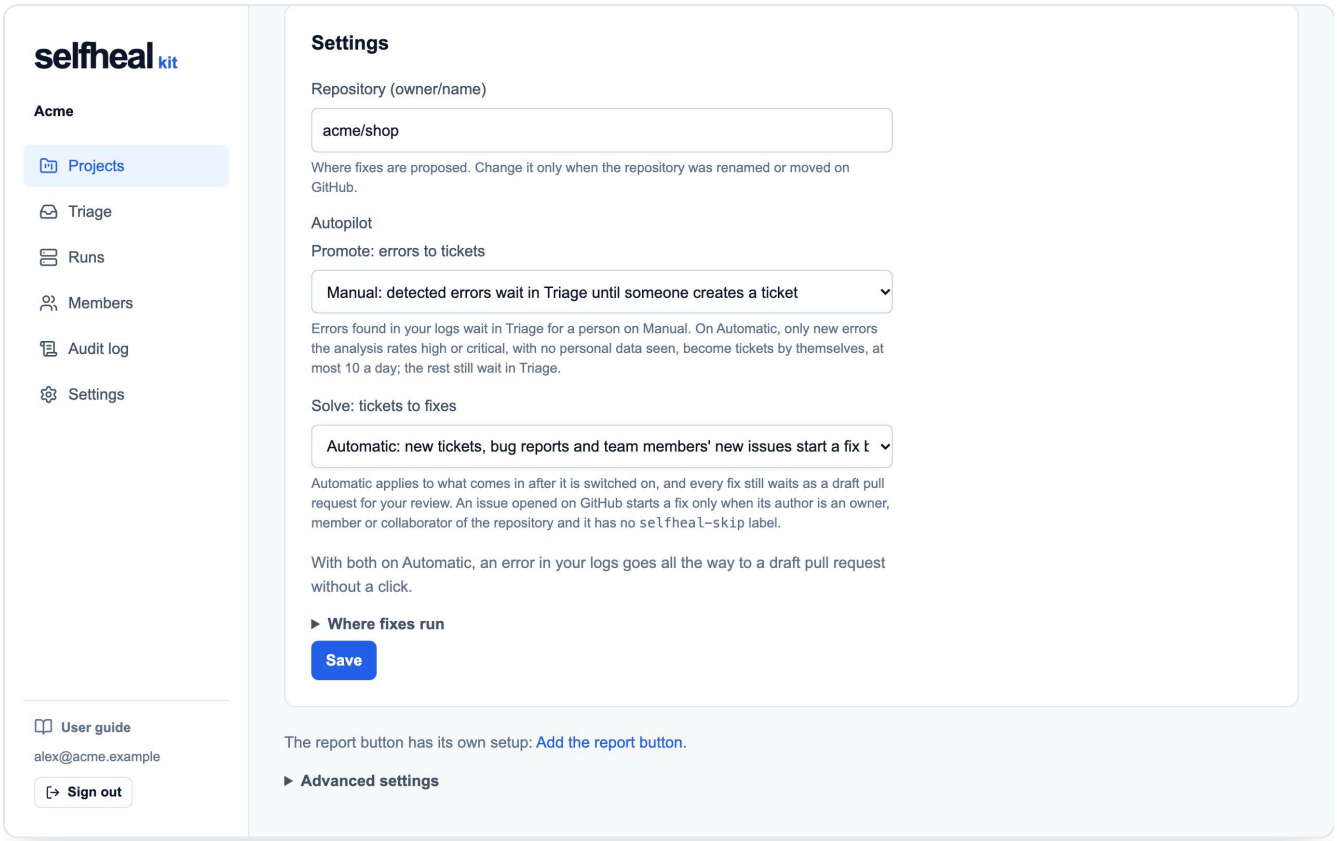
The setup's **Autopilot** asks two questions once, and the project's **Settings** keep the same two switches under **Autopilot**, each Automatic or Manual, shown on the project's page too:

- **Solve**, **Fix new bugs automatically? Off**: you start each fix yourself, with the **Fix** button or a member's `@claude` comment. **On**: new bug reports, team members' new issues and new tickets start a fix by themselves.
- **Promote**, **Turn serious errors in your logs into tickets automatically? Off**: errors found in your logs wait in Triage for a person. **On**: new errors the analysis rates high or critical become tickets by themselves, at most 10 a day; one where personal data was seen still waits for a person.

With both on, an error in your logs goes all the way to a draft pull request without a click; the pull request still waits for your review.



The question in the setup.



The same switch in the project settings.

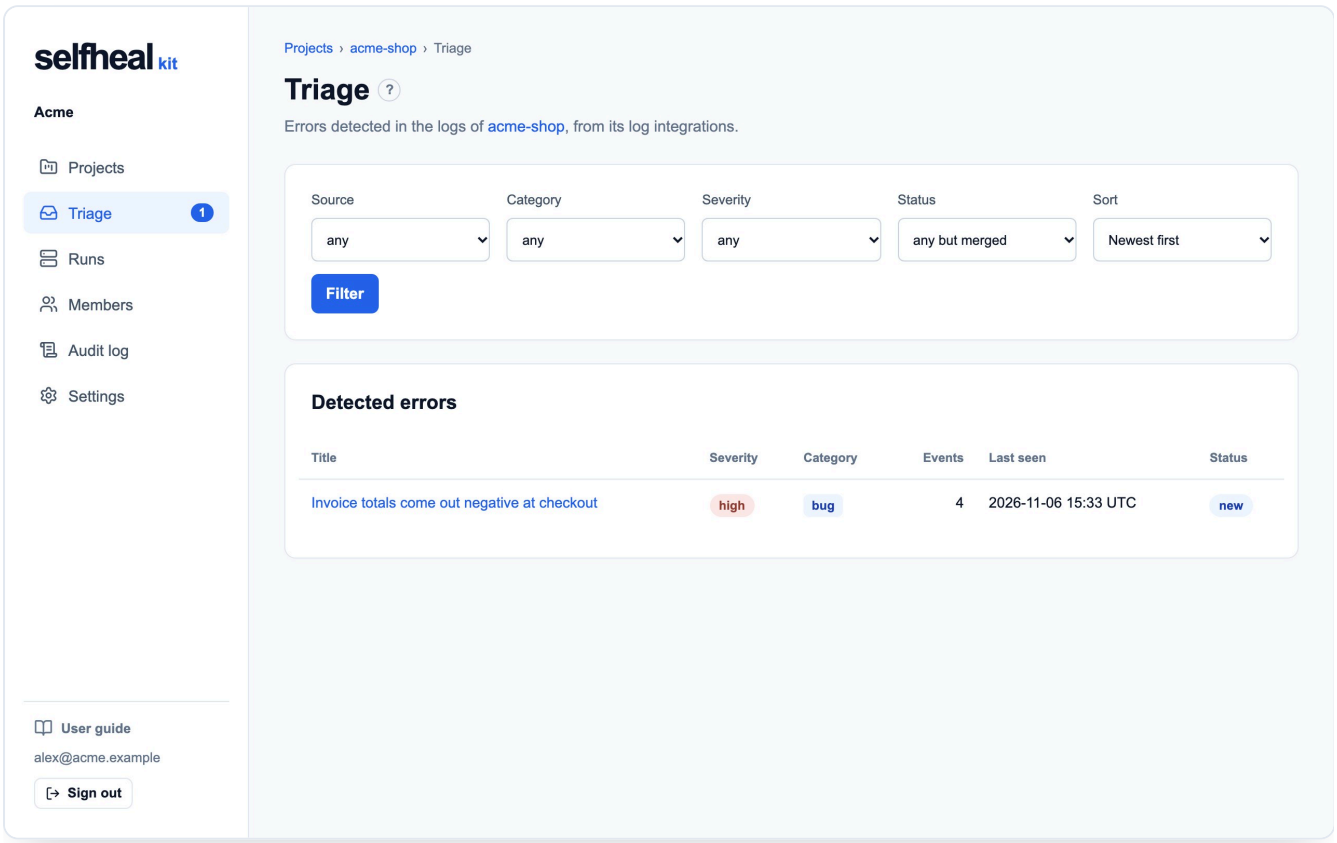
Good to know: it applies to bugs that come in after it is switched on. Whatever you choose, a fix always ends as a draft pull request that a person reviews. Under **Where fixes run**, you can choose Softup's AWS (nothing needed in your repository) or your repository's own GitHub Actions. For GitHub Actions, the project's **GitHub Actions** card lists what is left: the app's permissions, accepted on GitHub with **Review on GitHub**, then the pull request that adds the workflow, which you merge.

PART 2 · EVERYDAY USE

11. Triage: from error to ticket

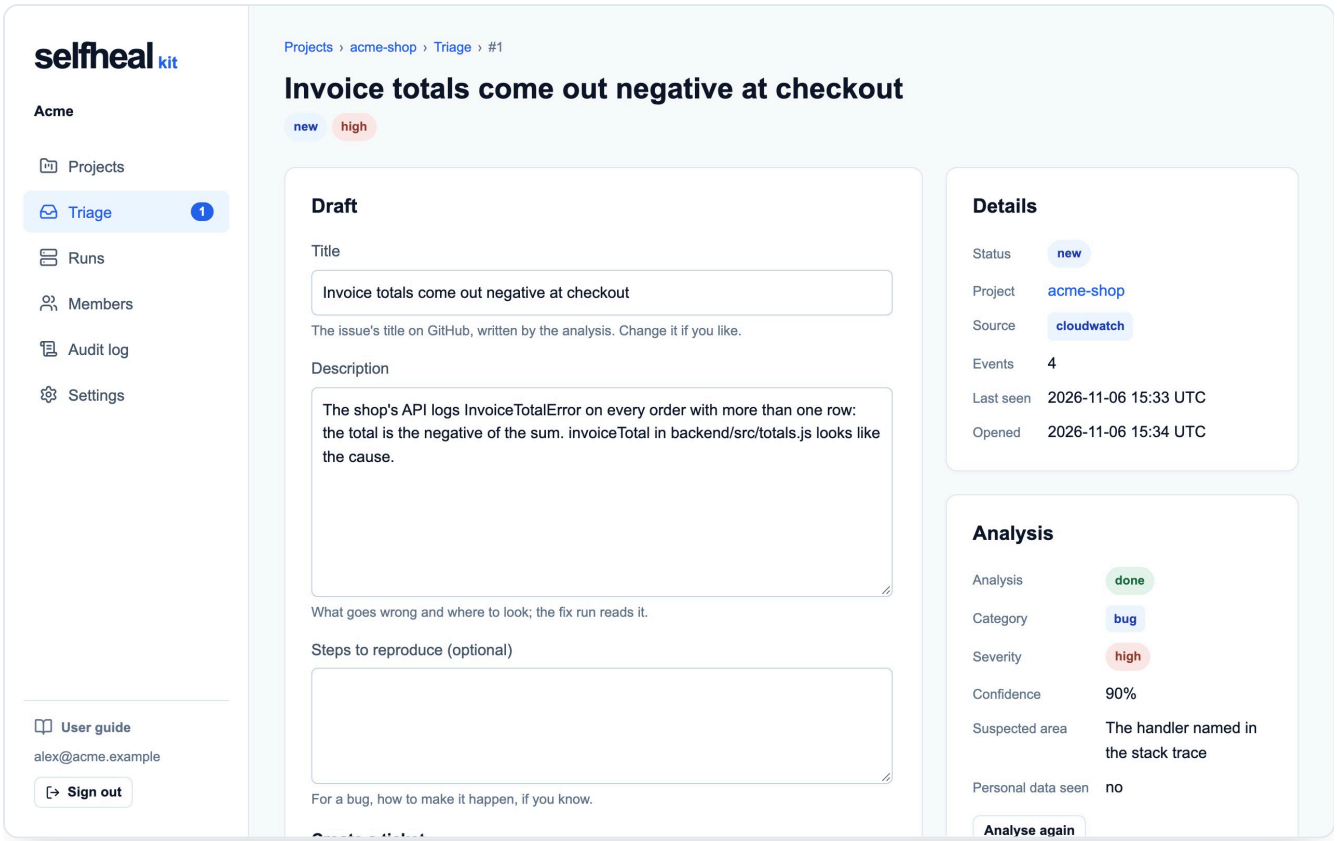
Who: everyone on the team

Errors from your logs do not become tickets straight away. They wait in **Triage** as detected errors, so your team decides what is worth fixing. Repeats of the same error are grouped into one, and Claude analyses each: its category, severity, confidence, and where in the code to look.

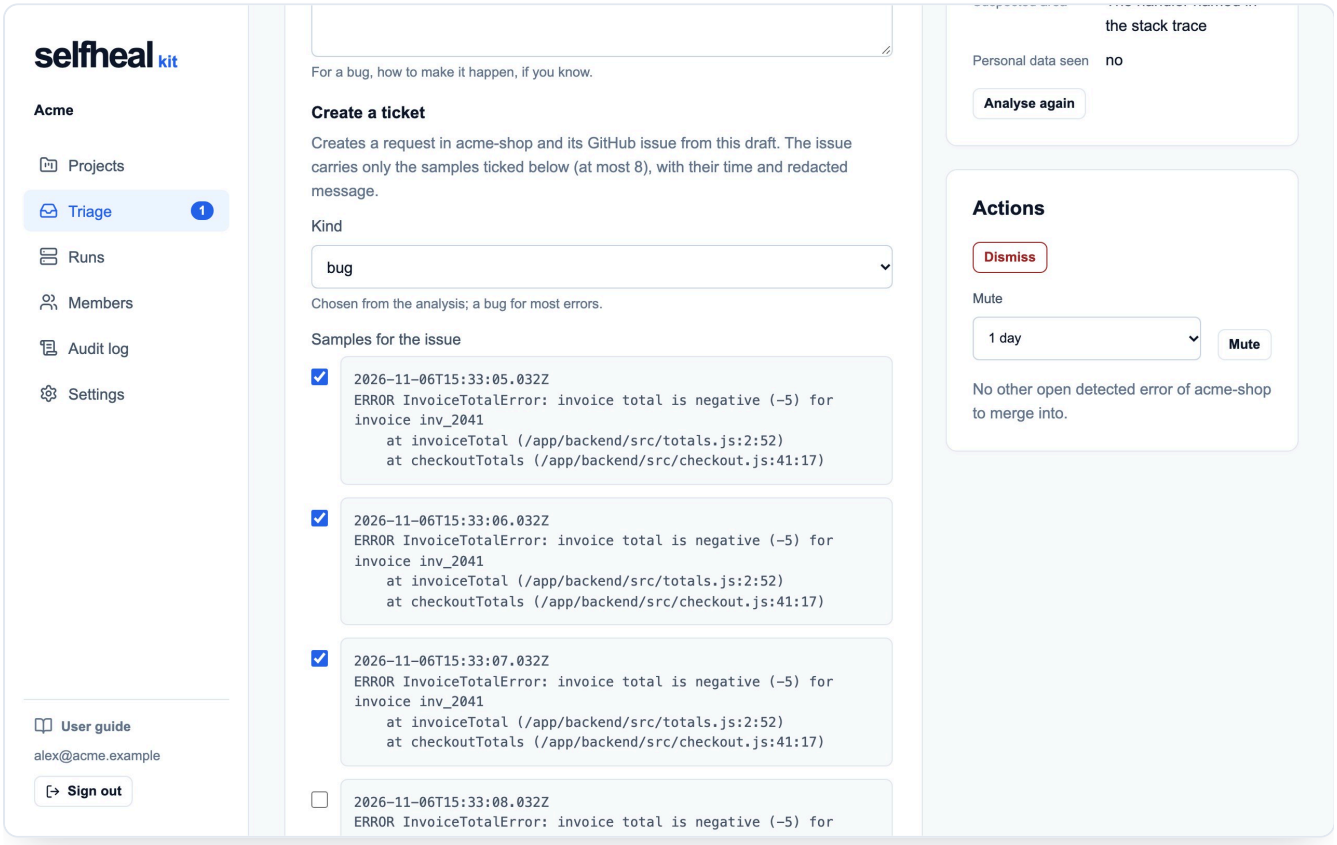


Triage lists the detected errors with their severity and number of events.

1. Open **Triage** in the menu and click a detected error.
2. Read the **Analysis** and the **Evidence**: the error's events over time and its redacted samples.
3. Adjust the draft's **Title** and **Description** if you like. The newest three samples are ticked to go into the issue; change the choice if you want (at most 8).
4. Click **Create ticket and fix**: the error becomes a request and an issue in your repository, and its fix starts at once. **Create ticket only** leaves the fix for later. When the project fixes new bugs automatically, there is one button, **Create ticket**, and the fix starts by itself.



A detected error: the draft on the left, the analysis on the right.



Creating a ticket, with one sample chosen.

Not worth fixing? **Dismiss** it (if it happens again, it shows up again), **Mute** it for a day, a week, a month or for good, or **Merge into** another detected error of the same problem.

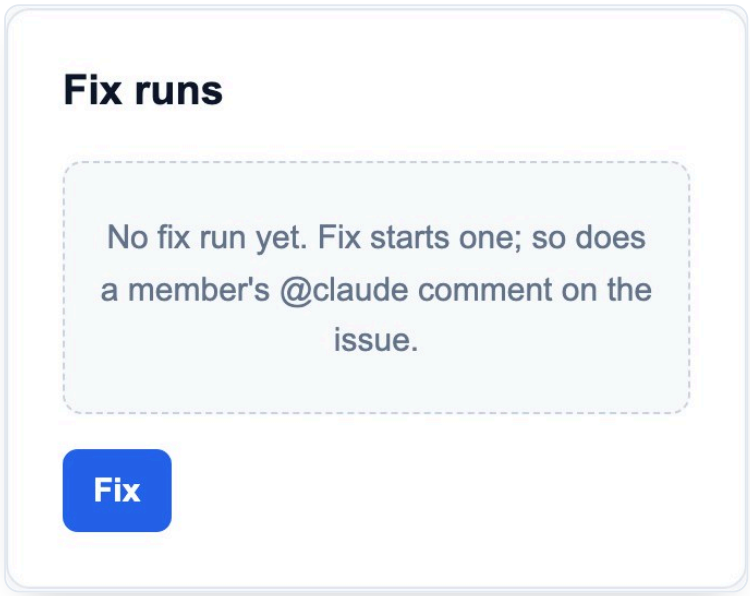
> **Check redaction:** if the analysis saw personal data the redactor may have missed, the error is marked. Read the evidence and click **I checked the evidence** before creating its ticket.

12. Start a fix

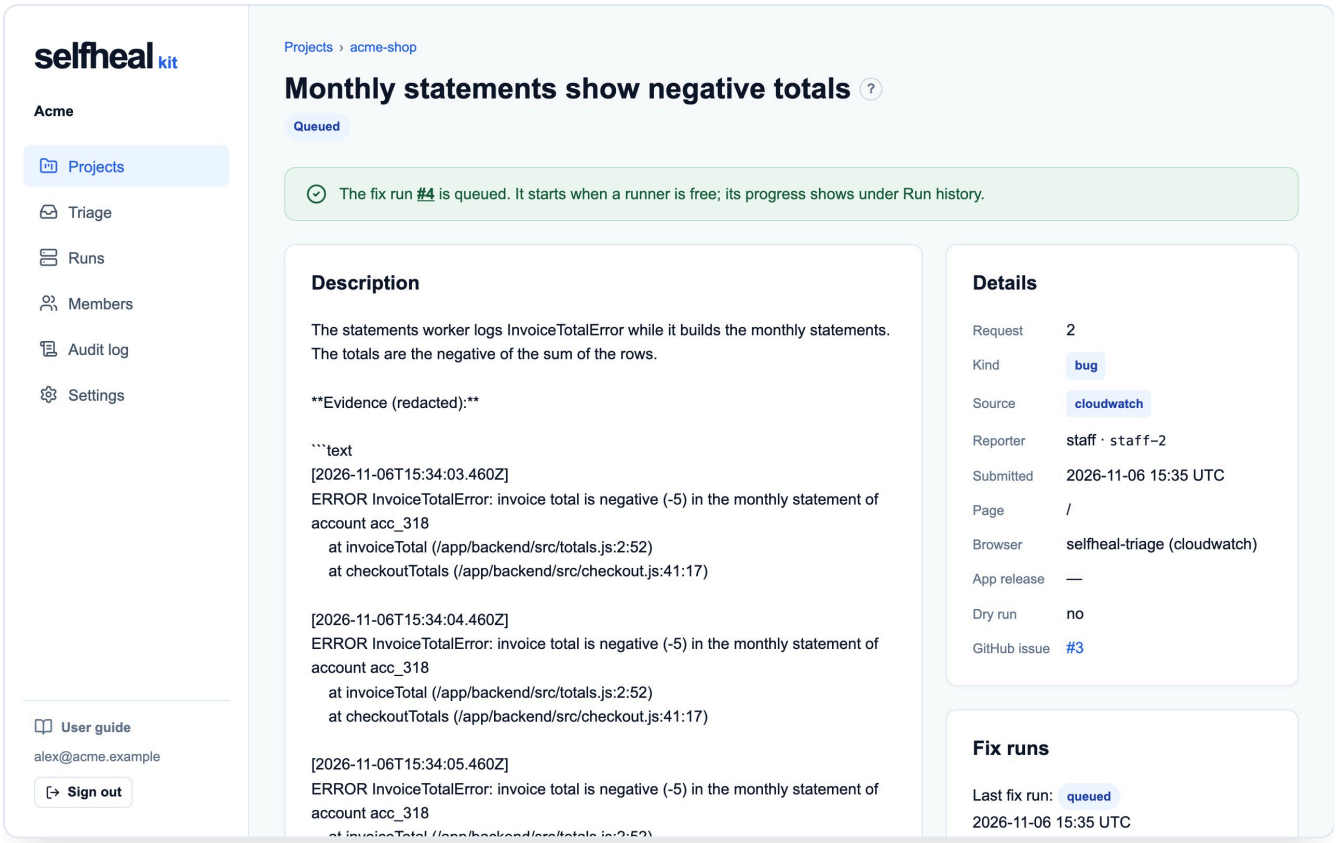
Who: everyone on the team

Every request has a page, opened from the project's list. When fixing new bugs automatically is off, a new request waits. Start a fix in either of two ways:

- Click **Fix** on the request's page, in the **Fix runs** card.
- Comment `@claude` at the start of a comment on the GitHub issue, optionally with instructions: `@claude the total should include tax`.



With automatic fixing off, the request waits for Fix.



After Fix: the run is queued.

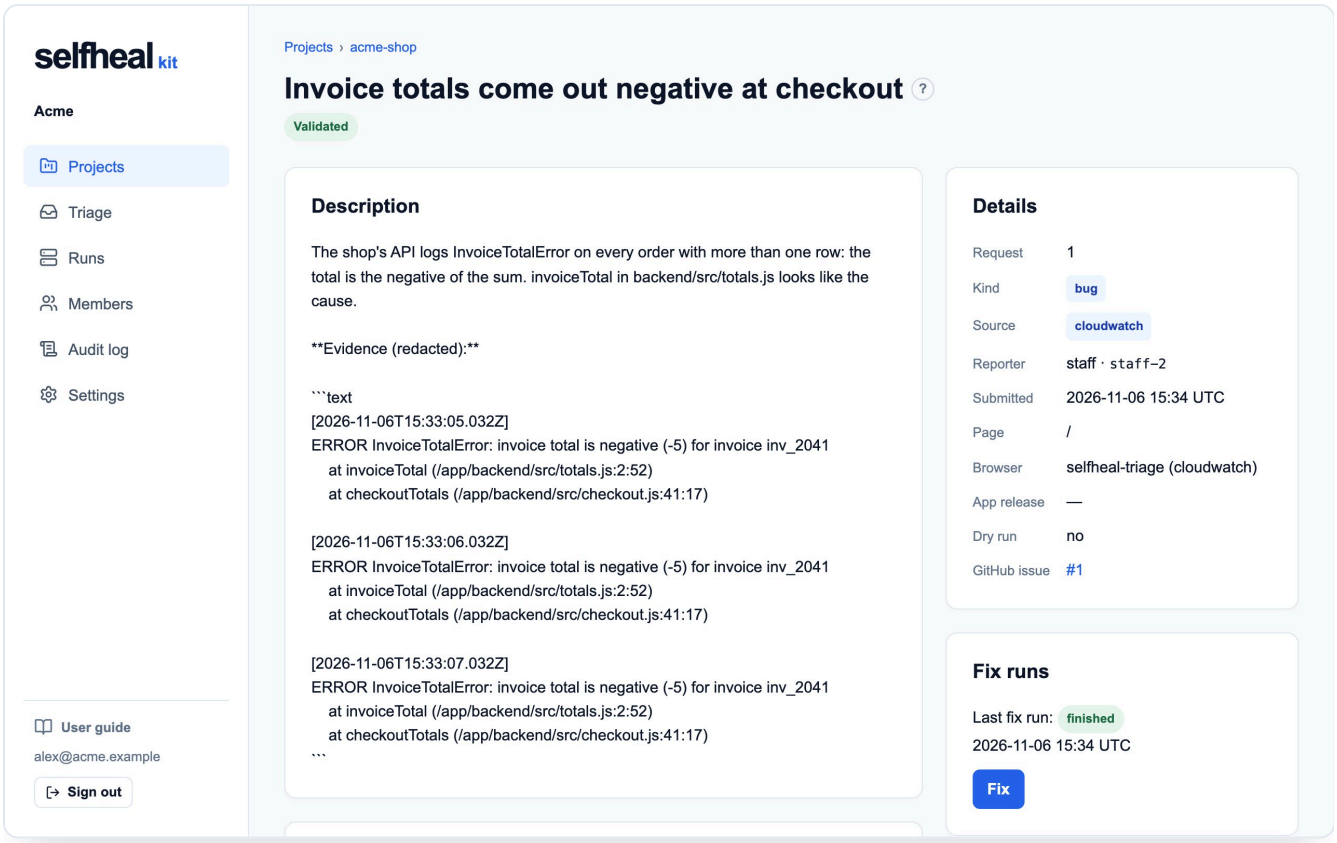
When it is on, the run has already started by the time you open the request.

> **Good to know:** only owners, members and collaborators of the repository can start a run from GitHub, and bots never can. A project allows 3 attempts per issue and 50 runs a month unless its profile says otherwise; when a limit refuses a run, the reason is posted on the issue.

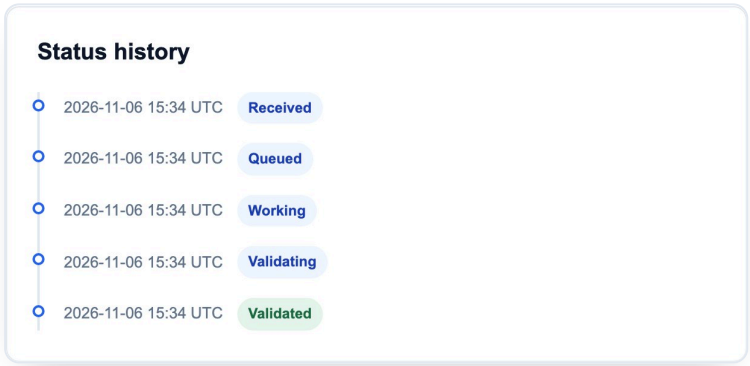
13. Follow a run and read its log

Who: everyone on the team

The request page follows the work from start to finish. Its **Status history** shows each step: the run is **Queued**, then **Working**; its draft pull request opens; a second run validates it (**Validating**); and the request ends **Validated**.

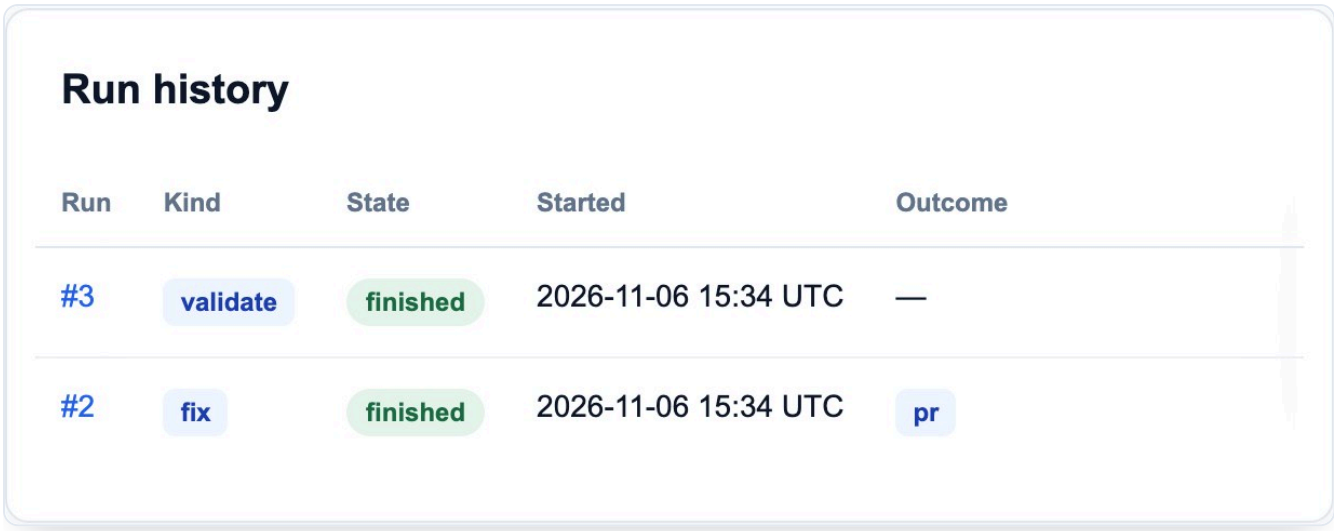


A request whose fix was validated.



The status history.

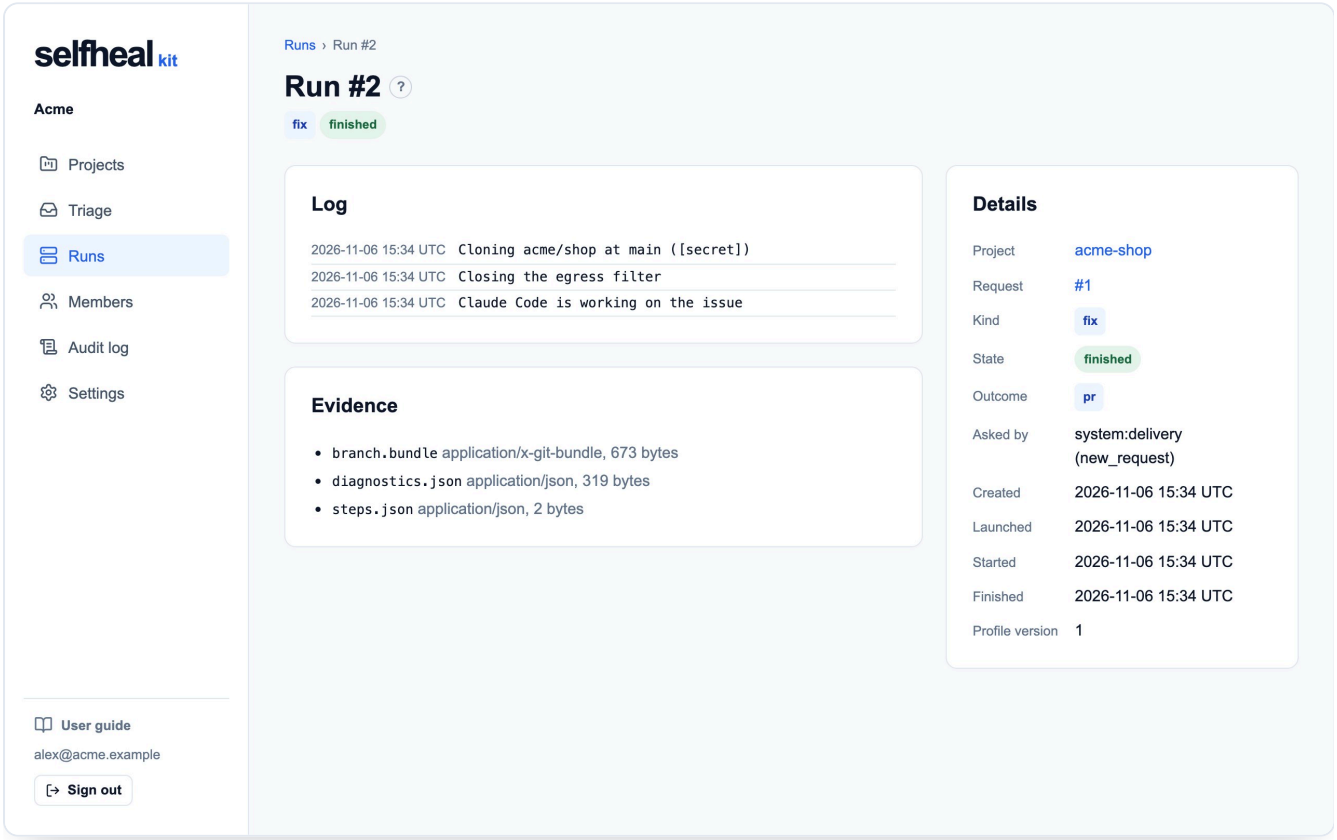
The **Run history** lists every run of the request: the **fix** run that wrote the change, and the **validate** run that tested it. Click a run to open its page.



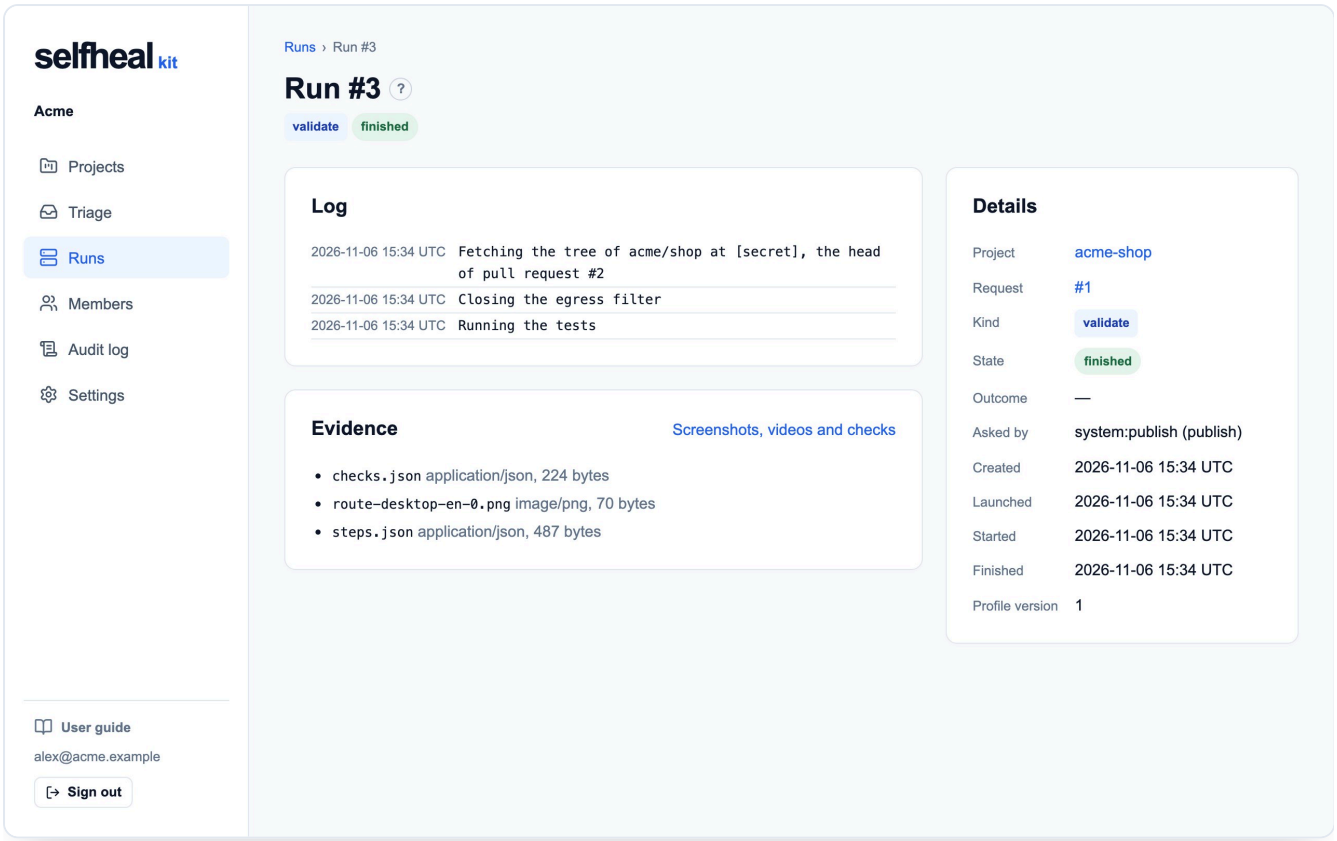
The runs of the request.

A run's page and its log

Each run has its own page. Its **Log** lists what the run did, step by step, with times; **Evidence** lists the files it kept; **Details** shows its state and outcome. An outcome of `pr` means the run produced a pull request.



The fix run: it cloned the repository, closed its network to everything it does not need, and let Claude Code work on the issue.



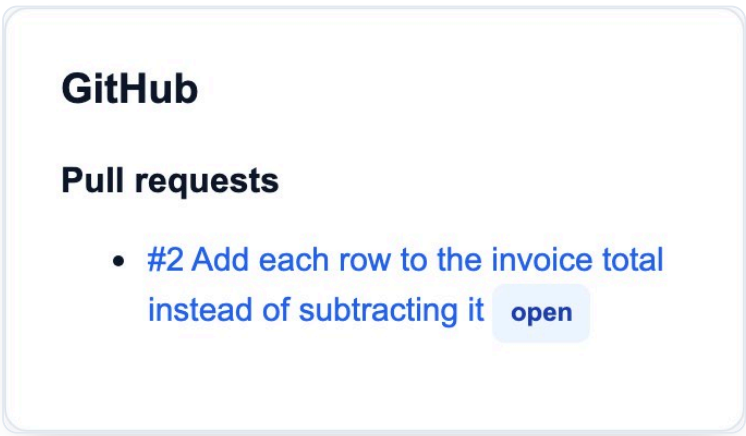
The validate run: it fetched the pull request's code and ran the project's tests.

> **Good to know:** a run works in a sealed environment: it reads your repository with a read-only token, and its network is closed to everything it does not need. Secrets never appear in the log.

14. Review the draft pull request

Who: everyone on the team

The fix arrives as a **draft pull request** on the branch `selfheal/issue-N`, linked from the request's **GitHub** card.



The draft pull request, linked from the request.

1. Check the **Validation** card: the checks that ran and whether they passed. On GitHub the commit status `Self-Healing UI validation` shows the same.
2. Open the pull request and read the change, as you would a colleague's.
3. Want something different? Comment `@claude` on the pull request or in a review, saying what to change. A new run updates the draft, and it is validated again.
4. Happy with it? Mark it ready for review, approve it and merge it on GitHub as usual.



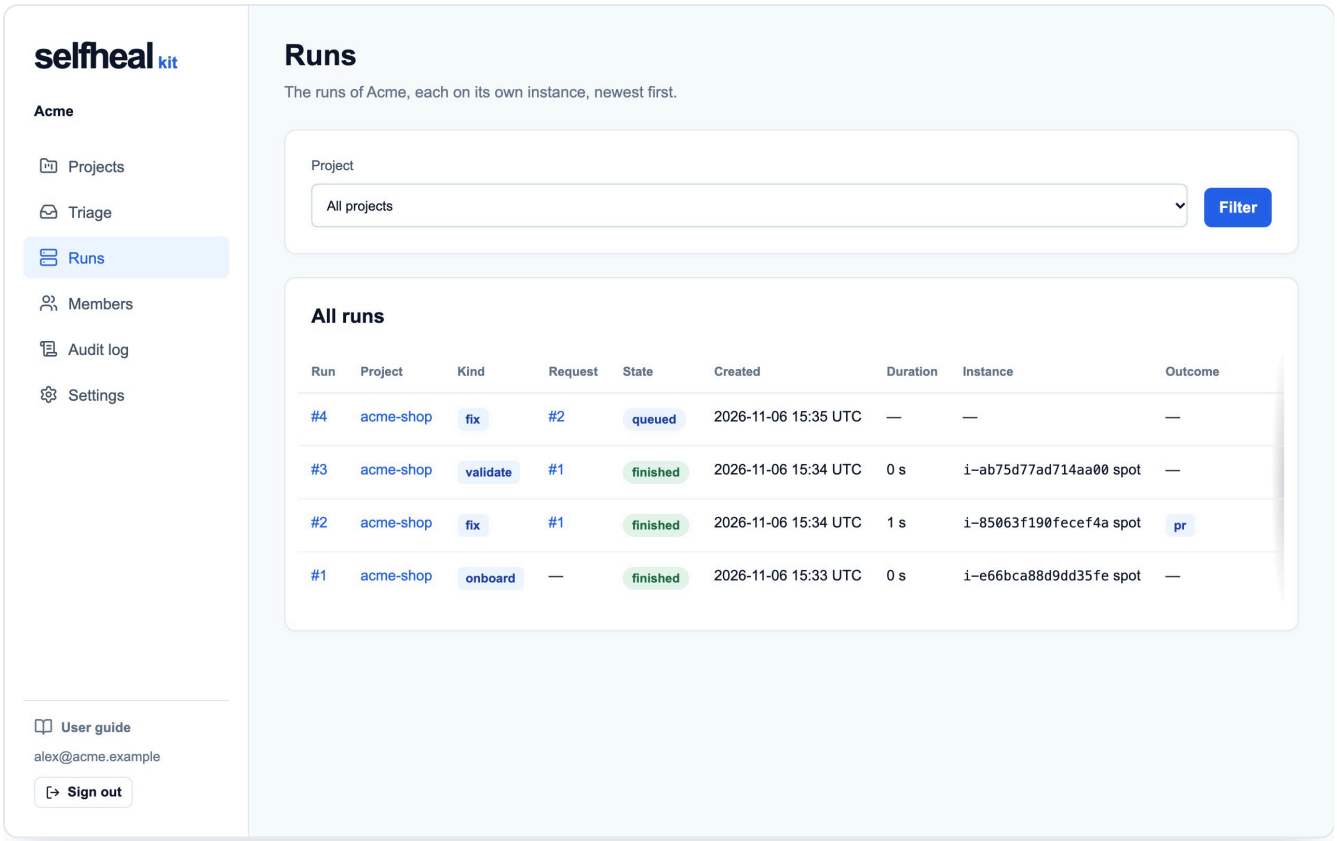
Validation: the fix passed the project's tests.

> **Good to know:** every new push to the draft is validated again. If a pull request is merged without a passing validation, the request page says so.

15. Runs, audit log, emails and limits

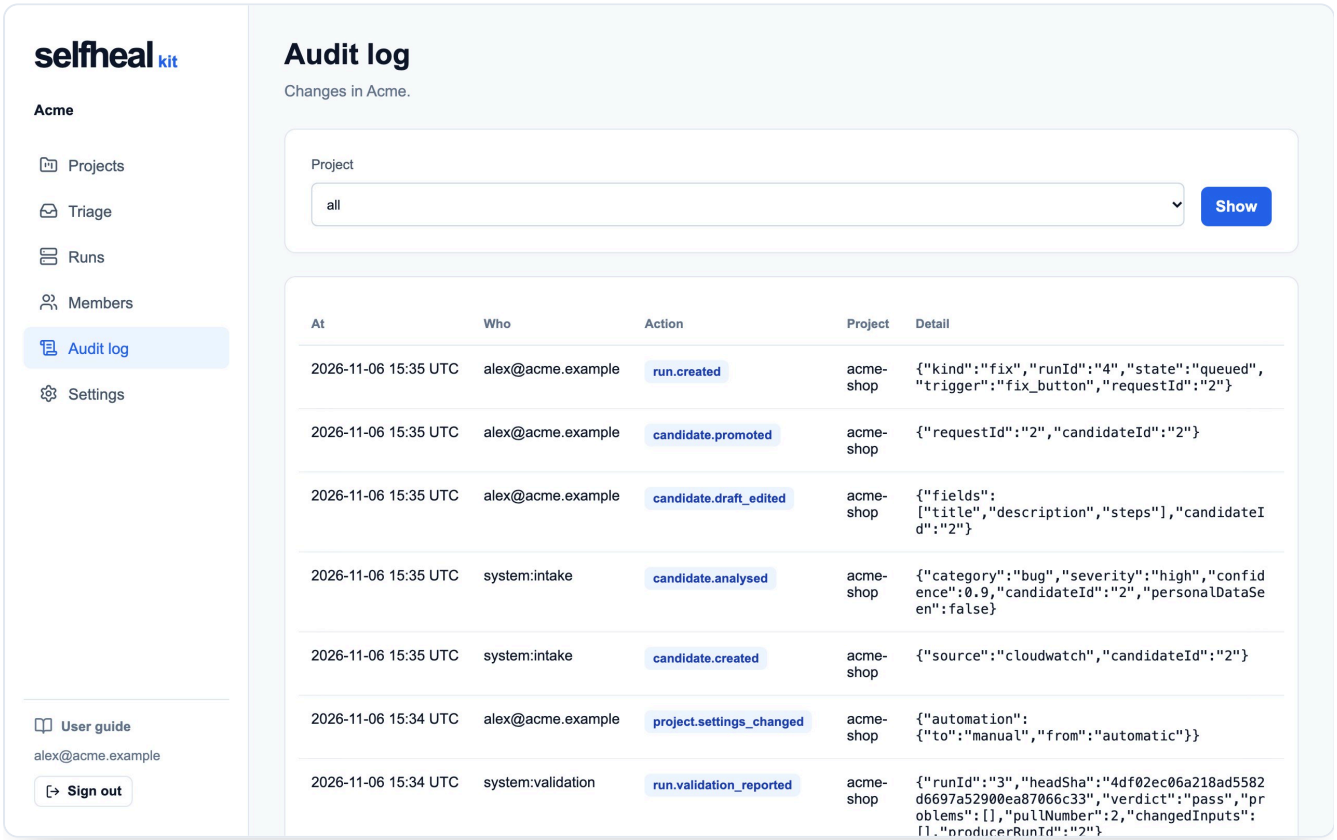
Who: owners and admins

Runs lists every run of the organization, newest first, with its project, kind, request, state and outcome.



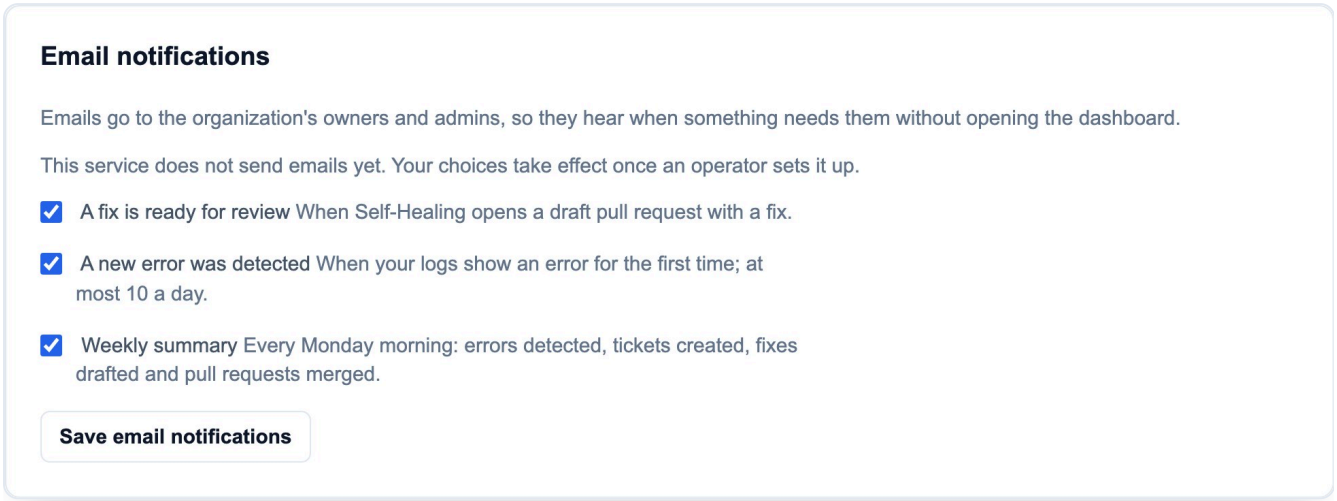
The organization's runs.

The **Audit log** records admin actions such as settings changes, key rotations and membership changes: who did what, and when.



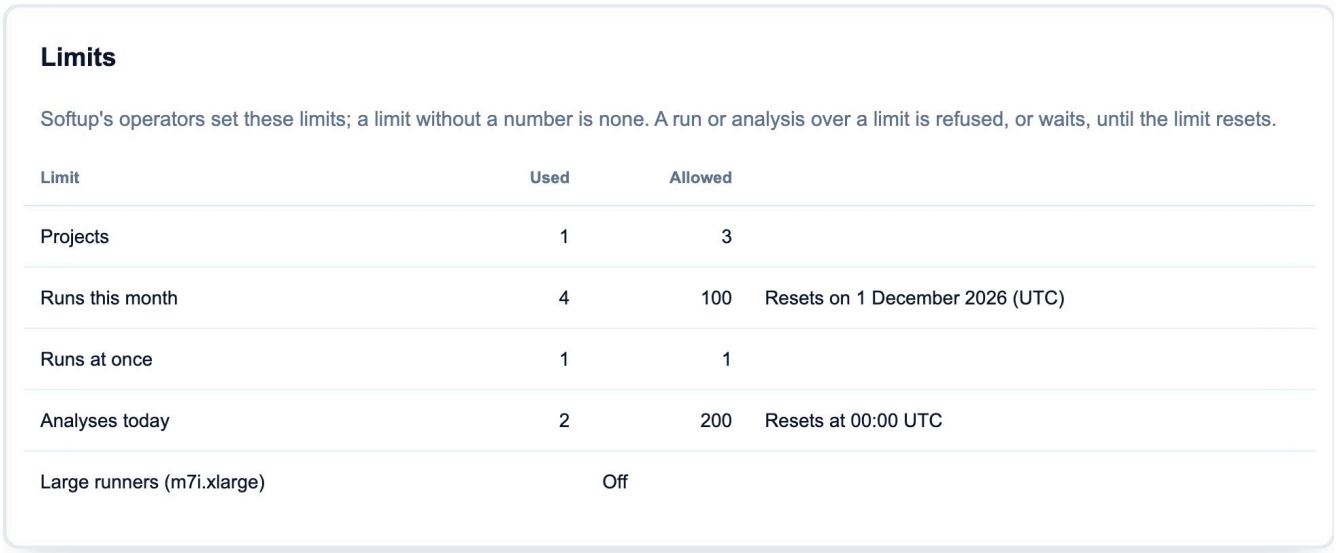
The audit log.

Owners and admins also hear from Self-Healing by email: when a fix is ready for review, when a new error is detected in your logs, and in a short summary every Monday. Each can be turned off in **Settings**, on the **Email notifications** card.



Choosing which emails owners and admins get.

The **Limits** card in **Settings** shows your organization's limits and what is used: projects, runs this month, runs at once and analyses today. Softup sets them; ask us if you need more.



Your organization's limits.

REFERENCE

16. Request statuses

Status	Meaning
Received	The request is recorded and its issue exists. With automatic fixing off, it waits for Fix or `@claude`.
Sending · Could not send	The issue is being created on GitHub, or could not be.
Queued · Working	A fix run is waiting for its turn, or working on the issue.
Draft PR open	The run opened a draft pull request.
Validating · Validated · Validation failed	A validate run is testing the draft; the tests passed; or they failed.
In review · Approved · Changes needed	The pull request is being reviewed, approved, or needs changes.
Answered	The run answered a question instead of changing code.
Run failed	The run could not finish or its result could not be published. The run's page says why.
Merged · PR closed	The pull request was merged, or closed without merging.
Completed · Not planned · Closed	The issue was closed on GitHub.

17. Roles

Role	Can
Member	See the projects and their requests, triage detected errors and create tickets, press Fix, and read runs.
Admin	Everything a member can, plus create projects, change project settings, the environment profile and integrations, manage Claude connections, invite and manage members, and see Runs and the Audit log.
Owner	Everything an admin can, plus invite owners, rename or delete the organization, and accept new terms.

18. When something is refused

You see	What to do
Invitation expired, used or for another address	Ask whoever invited you for a new invitation, and sign in with the Google account of the invited address.
No repository yet	Connect GitHub (step 1 of the setup, or Settings), choosing the repository.
A fix run refused: no Claude connection	Add an Anthropic API key: step 2 of the setup, or Settings, under Claude.
A fix run refused: no saved environment profile	Let Self-Healing learn the project (step 4 of the setup), or open the project's How it runs page and click Learn again.
A run refused by a limit	The reason is on the issue. Wait for the limit to reset, or ask Softup to raise it.
Creating a ticket waits for the redaction check	Read the detected error's evidence and click I checked the evidence.
A New terms banner	An owner accepts the new terms; until then admin changes wait.
Anything else	Write to info@softup.co.