

Kaloko help

How to set up Kaloko, walk your app with an agent, review what it built and keep a changelog and docs from the result. The pictures come from Kaloko's own accepted runs, so they show the screens as they are today; these pages are themselves living docs published with Kaloko.

Tutorials

- Your first run in ten minutes: install, sign in, walk a flow, share it and review it
- Design a flow with your agent: live HTML steps, versions and the design reference

Review and approve

- Review a run: go through the screens of a run on the canvas
- Approve or return a run: say yes to the steps that are right and send back the rest
- Review only what changed: review mode, its keys and approvals carried over
- Read what's new: the changelog with before and after pictures
- Hand over a signed acceptance record: proof of who accepted what, and how to check it

Walk and check

- Write a scenario: the flow, its steps and what each step must show
- Add criteria and check packs: automatic checks, questions for the evaluator, packs
- Walk in more browsers and on devices: WebKit, Firefox, phones, tablets, dark mode
- Walk Android, iOS and desktop apps: emulators, simulators, Electron, macOS and Windows
- Record a walkthrough video: recordings and traces of each step, one video of the run
- Run Kaloko in CI and comment on pull requests: walk each preview deployment
- Compare with a baseline and ignore what always changes: diffs without clocks and carousels
- Check pages on a schedule (hosted runs): Kaloko's browsers open your pages every day or week

Team and access

- Invite people and guests: members, guests on single projects, readers from your domain
- Roles and API tokens: who may do what, tokens for agents and CI
- Set up company sign-in: sign-in domains, two-step verification, SAML, OIDC and SCIM
- Connect Jira, Linear, Slack and Teams: issues from steps and channels per project

Changelog and docs

- Publish a changelog with pictures: draft the entry, record the release, share it
- Publish living docs and catch stale sections: guides whose pictures keep up with the app
- Export docs and the changelog: Markdown, Docusaurus, VitePress, MkDocs, HTML, PDF or JSON

Reference

- CLI commands: every command of kaloko
- Scenario file: every key of a scenario
- Criteria, checks and packs: every check type and what each pack adds

- Configuration file: kaloko.config.yml
- MCP tools: what agents can do over MCP

Explanation

- How Kaloko maps to your applications: organizations, products, projects, modules, environments and versions
- How verdicts are decided: checks, the evaluator and people
- Security and your data: what leaves your machine and who sees it

How Kaloko maps to your applications

One repository is rarely one app. A monorepo holds several apps, one app spans several repositories, an agency has a project per client, and every team names its environments differently. Kaloko uses a small set of concepts that you map to your situation once.

The concepts

| Concept | What it is | Typical mapping | |---|---|---| | Organization | your company or agency account | one per company | | Product | something released and versioned as a whole, with its own changelog and docs | a web shop, an admin, a mobile app, a package | | Project | a unit of work and access: who sees it and who reviews it | a client, a team, a module, a whole app | | Module | a part of a product with its own section in the docs and the changelog | checkout, account, billing | | Scenario and step | a flow and its screens | belongs to a project; may name a product and a module | | Environment | where a flow runs | local, preview, staging, production, production-cz | | Version | a released version of a product | from git tags, package.json, your release tool or kaloko release |

Products are optional

Without products, a project is the product and its changelog is the project's changelog. Most teams start there. When you need more, list products under products: in `kaloko.config.yml`, and scenarios name theirs with product: and module:. Nothing has to be moved.

A project can cover several products, and a product can be worked on in several projects. Docs and changelog entries refer to steps, so they know their product and module from the scenario.

Versions belong to products

An environment only says where a run happened. A run is tied to a product version, and the docs for version 2.4 show the accepted runs of 2.4. The pictures come from the environment the product names as its reference (`reference_env`), for example staging for an internal tool or production-cz for a Czech-only portal.

Four common set-ups

1. One app, one team. No products; the project is the app and has one changelog. This is the default.
2. A monorepo with several apps. One product per app; projects per team or per app.
3. An agency. A project per client and a product per deliverable. Client guests read only the changelog and docs of their product.
4. A large app with modules. One product with modules per team. Each team keeps its module's docs, and the release has one combined changelog with a section per module.

kaloko init proposes a product for each app under apps/ that has its own package.json. kaloko products sync puts them on Kaloko before their first release, and Products in the app's menu lists every product you read.

Picture: open the docs on Kaloko to see it



The Products page · en · desktop

Every piece works alone

You can use only the changelog, only the docs, only one module's docs, only the pictures for a docs site you already have (kaloko export), or only the "What's new" e-mail. Nothing requires the whole set.

How verdicts are decided

Every criterion on the canvas ends as pass, partial, fail or waiting for a person. Three kinds of judge take part, in a fixed order: checks that count, an AI evaluator for what needs judgement, and people, who have the last word.

Checks that count come first

Whatever can be measured is checked in the live page while the step is captured: a text is there, a field has a label, the address after the step matches, a colour equals a design token, the page loads under a time limit. Each result keeps its evidence, such as the element that matched or the measured value, so you can see why it passed or failed. These checks run without any AI and never cost an evaluator question.

An evaluator for judgement

Criteria such as "the error message tells the person what to do" go to a semantic evaluator. The default is JEV by TypeSafe. It reads a reduced outline of the page, not the screenshot, and returns a score between 0 and 1. Above the pass threshold (0.85 by default) the criterion passes, below the fail threshold (0.15) it fails, and in between it is marked for a person to decide. The thresholds are set in `kaloko.config.yml`.

Criteria marked vision: true, and answers in the unsure band, get a second look from a vision evaluator that sees the screenshot. It also says in one sentence what decided it; that sentence shows as "Why" on the canvas.

The evaluator runs with your own key in any plan, or through Kaloko from the Team plan up, within a monthly number of questions. Answers are kept, so evaluating an unchanged screen again asks nothing new. Without either, these criteria stay "not evaluated" until a reviewer decides. `kaloko calibrate` compares the evaluator with your reviewers' verdicts and suggests thresholds.

People decide

A reviewer approves a step, returns it with a note, or decides a criterion the evaluator was unsure about. At the end they accept or return the whole run. Every decision carries a name.

- Required approvers (Team plan and up): a run counts as accepted only after the named people approved it.
- Approvals carry over. When a new run is shared, a step whose screenshots and criteria did not change since the last accepted run, and whose checks still pass, keeps the approval a person gave it, with who and when. Only the rest waits in review mode. Anyone can revoke a carried approval. A carried approval never accepts the run and never counts for required approvers.
- Agents never approve for people. Decisions made by an agent or a token are marked as theirs and do not count as a person's.

Picture: open the docs on Kaloko to see it

Review mode · en · desktop

Why in this order

Counting is cheap and exact, so it goes first. Judgement is expensive and sometimes wrong, so it only handles what counting cannot, and it says when it is unsure. People see the evidence and the reason next to every verdict, and their decision is the one that is recorded as acceptance.

Security and your data

Kaloko runs where your code and your agent are: the CLI walks, captures and checks on your machine or in your CI. The service at kaloko.app stores what you share, shows the canvas and collects decisions. This page explains what travels where. Where the data is stored and which companies process it is on the trust page, which is kept up to date.

What leaves your machine

Nothing until you share. kaloko walk, evaluate and preview work locally, and the local canvas opens from disk. kaloko share uploads the run: screenshots, the captured HTML, criteria with their evidence, and recordings where your plan keeps them. Files travel by content hash, so a capture that did not change is not sent again.

What the evaluator sees

For text criteria the evaluator gets a reduced outline of the page with e-mail addresses and tokens masked, never the raw HTML. A screenshot goes to the vision evaluator only for criteria marked `vision: true` and for answers the text evaluator was unsure about. With your own key the request goes from your machine to that provider; without one, it goes through Kaloko.

Secrets stay out

Secrets are never written in `kaloko.config.yml`. The config names an environment variable of your choice (`password_env: STAGING_PASSWORD`), and the value is read from the environment, the project `.env` or `~/.config/kaloko/.env`. Values of those variables are removed from captured HTML, page text and console output before anything is stored. Sessions saved with `kaloko auth save` stay on your machine and are never uploaded.

Production is read-only

An environment marked as production is always read-only in the CLI: public pages as a visitor, no sign-in to back offices, no data created. A scenario must declare `readonly: true` to run there.

Runs expire

Every shared run has an expiry date, 30 days by default, at most your plan's retention, and is deleted after it. kaloko keep keeps a run for good; on Business and Enterprise accepted runs do not expire while the plan lasts. From the Team plan up you can download a run as a ZIP at any time with `kaloko export`.

How files are served

Uploaded files are served from a separate domain under short-lived signed links. Captured HTML is shown in a sandbox without scripts.

Who gets in

People sign in with a one-time e-mail link or with Google, Microsoft or GitHub, and can add a passkey or an authenticator app as a second step. Enterprise adds company sign-in over SAML or OIDC and SCIM.

Roles decide what a person may do: a Viewer reads, comments, approves and returns; a Designer or a Developer / Tester also uploads; an Admin also manages members, tokens and settings. Viewers are free in every plan. API tokens carry their own scopes and expiry, and the CLI's token from kaloko login lasts at most 30 days.

Picture: open the docs on Kaloko to see it



API tokens in Settings · en · desktop

Questions a security review asks

The trust page answers where data lives, how it is encrypted, the list of subprocessors and where the SOC 2 preparation stands. For a vulnerability report or the data processing agreement, write to the contact given there.

Hand over a signed acceptance record

When a client or an auditor asks who accepted a release, give them the acceptance record of the run. It lists who approved what and when, the results of the criteria and a fingerprint of every screenshot, and Kaloko signs it. Records are part of the Business and Enterprise plans.

Download it

1. Open the accepted run and its Acceptance record panel.

Picture: open the docs on Kaloko to see it



The acceptance record of an accepted run · en · desktop

2. Choose Download PDF for people, or JSON for a system that keeps records. The PDF carries the same JSON inside it.

A run that is not accepted has no record. Accept it first, or ask the person who has to.

Check one you received

Anyone can check a record without an account. Developers run `kaloko signoff --verify record.pdf`; it checks the signature against Kaloko's public key and says whether anything was changed. If a single character of the record was edited, the check fails.

Add criteria and check packs

Each step lists its criteria: what the screen has to meet. Kaloko knows three kinds, and a good scenario uses the cheapest one that can answer the question.

Countable: a deterministic check

Anything you can count or read from the page is checked in the page itself, with the evidence attached:

```
criteria:
- id: AC1
text: The cart shows two items.
check: { type: deterministic, assert: count, selector: ".cart-item", min: 2, max: 2 }
- id: AC2
text: Paying ends on the thank-you page.
check: { type: deterministic, assert: url_matches, pattern: "/thanks$" }
- id: AC3
text: The Pay button uses the brand colour.
check: { type: deterministic, assert: style_equals, selector: "button.pay", property:
background-color, token: color.brand.primary }
```

Other checks read text, attributes, the HTTP status, overflow on a phone or a value the page exposes (js_equals). The full list is in checks and packs. A step script never has to leave a marker for a criterion to find.

A question: a semantic check

What needs judgement goes to the evaluator as a question about the page:

```
- id: AC4
text: The primary call to action is clear.
check: { type: semantic, question: Does the page show one clear primary call to action?,
scope: main }
```

The evaluator reads a reduced outline of the page and answers with a score. A clear yes passes, a clear no fails, and anything in between goes to a person. Add vision: true to the check when the look is the point (layout, imagery, "looks like the brand"): the screenshot then gets a second look from a vision model. Keep it to the steps that need it.

A person decides: a manual check

```
- id: AC5
text: Legal approved the wording of the consent.
check: { type: manual }
```

A manual criterion stays "not evaluated" until a reviewer decides it on the canvas.

Check packs

A pack adds a fixed set of criteria to a step, so common checks are not written by hand:

```
nodes:
- id: home
packs: [seo, a11y, perf, console]
```

| Pack | What it checks | |---|---| | seo | one H1, title and description, canonical, hreflang, Open Graph, indexing, html lang | | usability | purpose, one primary action, link texts, unlabelled fields | | a11y | alt texts, names, labels, headings, contrast, landmarks, the Tab walk and visible focus | | perf | LCP, CLS, TTFB and page weight measured in the walk | | console | whether console errors touch what the step is for | | errors | error pages: a real 4xx or 5xx status, a way back, no stack trace | | app | app screens: labels on controls, touch target sizes, cut-off text, crashes | | design | an implementation against the accepted design: pixels, structure, tokens, contrast |

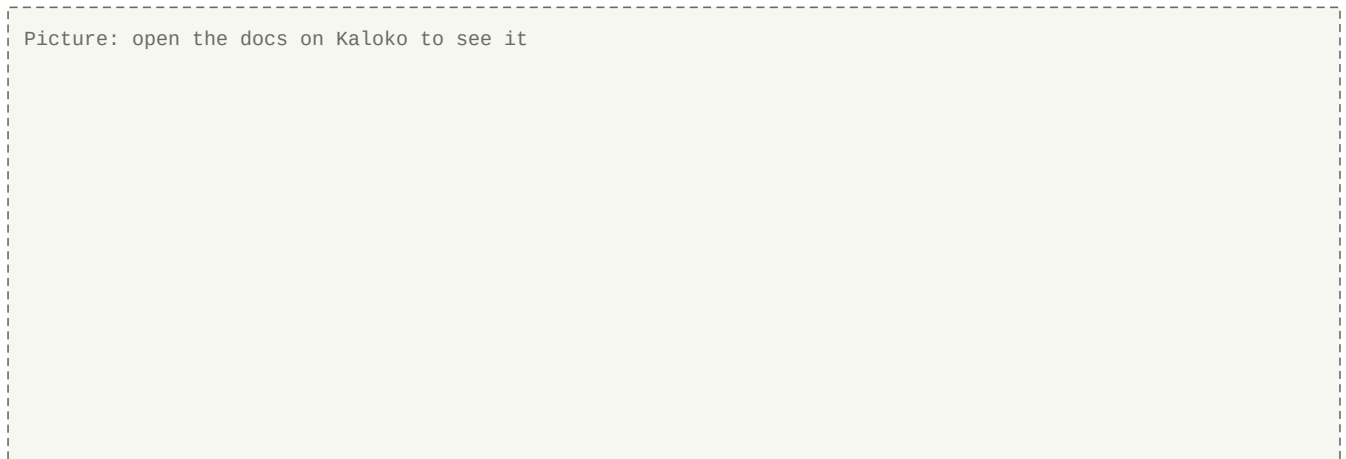
To change a limit, write your own criterion with the same id as the pack's (for example AC401 for LCP); yours wins. How verdicts come together is explained in how verdicts are decided.

Approve or return a run

Your decision is what turns "the agent says it is done" into "it is done". You decide step by step, then for the whole run.

Decide each step

1. Open a step and read its criteria. A green criterion was checked by Kaloko; one that needs a person says so, and that judgement is yours.



A step with its criteria and comments · en · desktop

2. Under the comments, choose Approve when the step is right. When it is not, write what should change and choose Reject: "the total jumps when the price loads", not "fix this". The agent reads the note and works from it.

Decide the run

1. When every step is approved, choose Accept run. The run becomes the picture of this version: the changelog and the docs show its screens from now on.

2. If something has to change first, choose Return run. Whoever shared it gets your notes; when they share the next run, the comment in the pull request points to it.



The run on the canvas · en · desktop

Next time is shorter

In the next run, steps whose screens did not change keep your approval. You only look at what is new or different, and the canvas says which approvals were carried over and from which run.

Agents can comment and suggest, but only a person accepts a run.

Walk in more browsers and on devices

By default a run captures every step in Chromium, in each locale and viewport of the scenario. You can add browser engines, device presets, dark mode and a few more settings. Every combination becomes its own capture, so add only what the task is about: a Safari bug needs WebKit on the affected steps, not every setting on every step.

In the scenario

```
viewports: [iphone-se, iphone-15, pixel-8, ipad-landscape, laptop]
browsers: [chromium, webkit, firefox]
color_schemes: [light, dark]
reduced_motion: [no-preference, reduce] # optional
forced_colors: [none, active] # optional, Windows high contrast
display_modes: [browser, standalone] # optional, the page as an installed web app
```

Defaults for the whole project go under walk: in kaloko.config.yml. The command line wins over the scenario, and the scenario over the config.

For one run

```
npx kaloko start --scenario qa/flows/checkout.yml --env staging \
--browsers chromium,webkit --color-schemes light,dark --viewports
iphone-se,pixel-8,desktop-hd
npx kaloko walk # one pass per engine and device
npx kaloko walk --browsers webkit # only the WebKit captures of the run
```

Device presets

npx kaloko devices lists the presets: phones (iphone-se, iphone-15, pixel-8, galaxy-s24...), foldables (fold-folded, fold-open), tablets (ipad-portrait, ipad-landscape...), computers (laptop, desktop-hd, desktop-wide) and tv-1080p. Add -landscape to turn a phone, foldable or tablet sideways (iphone-15-landscape). A preset sets the size, pixel ratio, touch and the device's user agent. It catches layout bugs; a real device catches the rest.

Install the engines

WebKit (Safari's engine) and Firefox are downloaded once per machine or CI job:

```
npx kaloko browser install --browsers webkit,firefox
```

npx kaloko doctor lists the engines your scenarios need and says how to install a missing one. On Linux CI add --with-deps for the system libraries.

What to expect

- A few checks run only in Chromium: names from the accessibility tree, visible focus and layout shifts, and on WebKit also the Tab walk. On other engines they say "not evaluated" with the reason and do not make the step partial.
- Contrast is checked per colour scheme, so a dark-mode contrast bug fails on the dark capture.
- A run refuses more than 400 combinations.

On the canvas the browser, colour scheme and other settings sit next to the language switcher. The step detail of a device preset draws the device frame, and Beside: shows the same step in another browser or scheme next to it.

Set up company sign-in

For organization admins. Kaloko works with the sign-in your company already uses: personal sign-in on every plan, an organization-wide second factor from Business, and your identity provider with SCIM on Enterprise.

Sign-in on every plan

People sign in with a one-time link sent by e-mail, or with Google, Microsoft or GitHub. Each person can add a passkey or an authenticator app as a second step in their account settings.

Picture: open the docs on Kaloko to see it

The sign-in page · en · desktop

On Business an admin can require the second step for the whole organization and set limits for sessions, under Settings -> Security. People without a second step are asked to add one before they continue.

Verify your domain first

Company sign-in and SCIM act only on addresses from verified domains.

1. Open Settings -> People and add your domain under Sign-in domains.
2. Add the TXT record Kaloko shows to your DNS, or prove the domain by signing in with Google Workspace or Microsoft 365.
3. Choose Verify now. Until the domain is verified, nobody joins through it.

Connect your identity provider (Enterprise)

1. Open Settings -> Security -> Company sign-in.
2. Choose SAML (Okta, Entra ID, Google Workspace and others) and paste your provider's metadata, or OIDC with the issuer, client ID and secret. Kaloko's SP metadata, with its certificate, is on the same page.
3. Choose Test connection. It shows what the provider sent and whether Kaloko would accept it; nobody is signed in by the test.
4. Enforce it. People from your verified domains then sign in only through your provider. Admins with a passkey or an authenticator app keep a way in for the day the provider is down.

Optional settings on the same page:

- Encrypted assertions: create the organization's key pair; a new pair keeps the old one working until you remove it.
- Single logout: signing out at the provider signs people out of Kaloko, and the other way round.
- Sign-in from the app tile of your provider, off until you turn it on, with the page people land on.
- Your provider's second step can count as Kaloko's, for sign-ins through your provider.

Guests from other companies keep signing in as before and see only the projects they were invited to.

Provision people with SCIM (Enterprise)

Paste the SCIM base URL and token from Settings into Okta or Entra ID. Assigned people join with their domain's default role, pushed groups map to projects and roles, and removing someone ends their sessions, revokes their tokens and takes them off every project. SCIM never makes anyone an admin.

Limit where people come from (Enterprise)

Under Settings -> Security you can allow IP ranges for the app and, separately, for the API, the CLI and MCP. The audit log can be streamed to a signed webhook, Splunk or a Datadog-compatible endpoint.

Where your data lives and who processes it: Security and data.

Compare with a baseline and ignore what always changes

Before a release you want one answer: what looks different from the version the team accepted, and is it on purpose. A good run of a scenario becomes its baseline, and every later run is compared with it.

Set the baseline

On the canvas, open a run you are happy with, choose Accept run and then Set as baseline. Each scenario has one baseline, and it moves only when someone sets a new one. From the terminal:

```
npx kaloko reference --scenario checkout --baseline --run <run-id>
npx kaloko reference --scenario checkout # show the current one
```

Compare a run

On the canvas, choose Compare and the baseline (or the previous run, or any other run of the scenario). Each step gets a badge with the share of pixels that changed, and criteria whose verdict or evidence changed are marked.

Picture: open the docs on Kaloko to see it

A run compared with its baseline · en · desktop

In the step detail you can look at the change four ways: Side by side, Diff, Swipe (drag a line across the two screenshots) and Onion skin (fade one into the other). G shows one step in every language and screen size at once.

The same comparison as text, for an agent or the release notes:

```
npx kaloko compare # the current run against the baseline, else the previous run
npx kaloko compare --run <id> --baseline <id> --json
```

Ignore what always changes

Clocks, ads, carousels and "3 minutes ago" change on every run and drown the real differences. Leave them out of the pixel comparison with ignore: on the step:

```
nodes:
- id: home
ignore:
- { selector: ".ticker", note: live prices }
- { rect: [0, 620, 1280, 180], viewport: desktop }
```

You do not have to find them by hand:

```
npx kaloko walk --stability 3 # three screenshots of each capture a moment apart
npx kaloko stability # what moved, compared with earlier runs of the scenario
```

```
npx kaloko stability --apply # writes the suggestions into ignore:
```

Look at the screenshots before you apply them: a region that changed because the page really changed is not noise.

Reviewers can mark noise too. In the step detail on kaloko.app, Mark a changing region and draw a rectangle on the screenshot. It shows dashed on every run of the flow, and the canvas diff skips it at once. To put those regions into the scenario, run:

```
npx kaloko stability --pull
```

Walk the affected steps again afterwards: a capture records the regions it ignored.

Connect Jira, Linear, Slack and Teams

For organization admins on the Team plan and up. Once connected, people file a returned step as an issue straight from the canvas, and each project tells its own chat channel what happened.

Jira or Linear

1. Open Settings -> Integrations and connect Jira Cloud (sign in with Atlassian, or paste an API token) or Linear (sign in, or paste an API key).
2. Map each Kaloko project to a Jira project and issue type, or to a Linear team. A default row covers projects you do not list.
3. For Jira, add the webhook Kaloko shows to your Jira site, so a finished issue closes its comment in Kaloko at once. Without it, Kaloko checks every 30 minutes.

People then choose New issue on a step. The issue gets the screenshot, the criterion and the reviewers' notes, with a link back.

Slack or Teams channels

1. In Settings -> Integrations, add a channel for a project: Add to Slack for Slack, or the address of a Teams Workflows flow for Microsoft Teams.
 2. Tick the events the channel wants: new run, returned, accepted, comment, mention, release recorded.
 3. Set quiet hours if the channel should not hear from Kaloko at night; messages wait until they end.
- A restricted project only reaches channels that name it. The organization-wide Slack channel keeps working.

Export docs and the changelog

Your docs site may already have its own build, or a client wants the guide as a PDF. kaloko export writes the docs and the changelog with their pictures as plain files, so nothing stays locked in Kaloko.

Pick a format

```
npx kaloko export --format docusaurus --out website/docs
```

| --format | What you get | Plan | |---|---|---| | md | Markdown with an assets/ folder of pictures, for any static site | every plan | | json | the changelog as structured JSON | every plan | | docusaurus | a Docusaurus folder with front matter, categories and sidebars | Business and up | | vitepress | a VitePress folder with its config | Business and up | | mkdocs | a MkDocs folder with mkdocs.yml | Business and up | | html | offline HTML: a page per guide, the changelog and a print page | Business and up | | pdf | one PDF, printed by the browser | Business and up |

The step references become ordinary image files, in the language and screen size of the export.

Choose what goes in

- --what docs or --what changelog exports only one of them; the default is both.
- --version 2.4.0 exports the docs of a release instead of the latest.
- --lang cs exports one language.
- --product shop picks a product when the config has several.
- --out names the folder or file to write.

Published or local

By default the export takes what was published with kaloko docs publish. --local takes the files in your repository instead, without publishing them, which is handy for checking a change before it goes out.

Keep a docs site in step

An existing site keeps its own build. Either export into its folder and commit the result, or run the export in the site's CI before it builds:

```
npx kaloko export --format vitepress --what docs --out docs-site/guide
```

The token in CI needs only read access; see Roles and API tokens.

Download from the docs page

Readers who prefer a file can use Download on the docs or changelog page. It offers the same formats; on plans below Business only Markdown and JSON are available there too.

Every export opens without Kaloko. For the source of the pages and how to keep them current, see Publish living docs and catch stale sections.

Check pages on a schedule (hosted runs)

The marketing site changes several times a week, and none of it goes through your pull requests. A hosted run opens the pages of a read-only scenario on Kaloko's own browsers every day or week, takes full-page screenshots in every language and screen size, checks what can be read from the page and shares the run under your name. Nobody has to start the CLI. Hosted runs are part of the Business plan (2,000 loaded pages a month) and Enterprise (10,000).

1. Prepare a read-only scenario

The scenario's steps open pages. To start from the pages themselves:

```
npx kaloko draft https://example.com/ https://example.com/pricing
```

2. Schedule it

```
npx kaloko schedule add --scenario qa/flows/public.yml --env production --every day --at 5
npx kaloko schedule add --scenario qa/flows/public.yml --env production --every week
--weekday mon
```

--at is the hour in UTC. --every manual creates a schedule that runs only when you start it.

Pages behind a gate work too. HTTP Basic credentials of the environment are sent once and kept encrypted. For signed-in pages, a person saves a session with `npx kaloko auth save --env production --account member`, and the schedule uses it with `--account member`.

3. Manage it

```
npx kaloko schedule list # next and last run, pages used this month
npx kaloko schedule run <id> # run it now
npx kaloko schedule pause <id>
npx kaloko schedule resume <id>
npx kaloko schedule remove <id>
```

Settings -> Integrations on kaloko.app shows the same schedules.

Read the result

The run appears among the others, compared with the scenario's baseline, with the same canvas, comments and approvals. Checks of selectors, texts, titles and meta tags are decided. Criteria that need the full walk say "not evaluated" and give the reason.

What hosted runs cannot do

They only open pages and read them. Nothing is clicked, typed, submitted or created, so steps that do more than open a page are left out and named. Native apps, local or private addresses, credentials sent as headers and base URLs per language also need the full walk.

For those, export the schedule as a GitHub Actions workflow that walks the scenario with the full CLI:

```
npx kaloko schedule export github --scenario qa/flows/checkout.yml --env staging --every week
```

The workflow names the secrets it needs.

How pages are counted

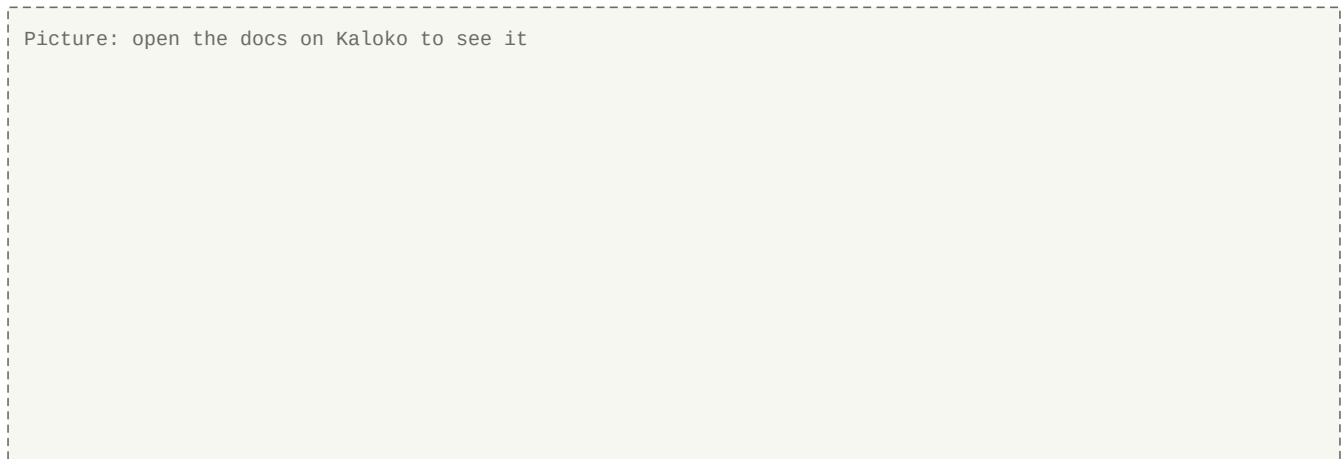
Every loaded page counts: each step in each language and screen size. A scenario of 10 pages in 2 languages on desktop and mobile loads 40 pages per run, so a daily run uses about 1,200 pages a month.

Invite people and guests

A run is only useful once the people who decide have seen it. Bring colleagues into the organization, invite a client to a single project, or let everyone from your company domain read along. Viewers are free on every plan, so nobody has to ration who may look.

Invite a colleague to the organization

1. Open Settings -> People and choose Invite to the organization.



Members in Settings · en · desktop

2. Enter the e-mail address and pick a role. A Viewer reads, comments and approves; the other roles are explained in Roles and API tokens.

3. The person gets an e-mail with a link and joins the organization with it.

The list in Settings shows every member with their role and when they were last seen, and pending invitations with who sent them and when they expire.

Invite someone to one project

Clients, freelancers and colleagues from other teams often need one project, not the whole organization.

1. Open the project and choose Share.

2. Type the e-mail address, also one outside your organization, and pick a role.

3. Choose Invite. The invitation is valid for 14 days.

An address outside your organization joins as a guest. Guests see only the projects they were invited to: not the other projects, not the people of the organization, not settings or billing. The Share dialog also shows who already has access and whether the project is open to every member or restricted to the people listed.

On Business you can set a date when a guest's access ends; on that day they leave the project and get an e-mail. Guests with the Designer or Developer / Tester role are part of Enterprise; on other plans a guest is a Viewer.

Let your company domain sign in

Instead of inviting colleagues one by one, add your company domain under Settings -> People -> Sign-in domains. Prove you control it with the TXT record Kaloko shows; until the domain is verified, nobody joins through it. Anyone with an e-mail on a verified domain can then sign in with the domain's default role (Viewer unless you choose another). Public mailbox domains such as gmail.com cannot be claimed.

Readers from your domain

On Business, the Readers from our domain switch in the same place lets everyone who signs in from a verified domain read the changelog and docs of every product, also where the project is restricted. They join as Viewers and take no seat. See Publish a changelog with pictures.

Who takes a seat

Viewers never do. Designers, developers / testers and admins take one seat each, once for the whole organization, members and guests alike. An invitation with such a role holds a seat until it is accepted or revoked.

Publish a changelog with pictures

Your accepted runs already hold every screen of a release. Kaloko turns CHANGELOG.md in your repository into a page with before and after pictures that clients, support and marketing can read. Kaloko's own changelog is made this way and is public: kaloko.app/p/sinfin/kaloko/changelog.

Set it up once

```
npx kaloko init --docs=changelog
```

It adds CHANGELOG.md (Keep a Changelog) and the changelog block in kaloko.config.yml. A repository with several apps lists them under products;; npx kaloko products sync puts them on Kaloko before their first release.

Put pictures in the entry

A picture names a step instead of a file:

- The pricing page compares every plan in one table.

```
![Before](kaloko:KALOKO-public/pricing?version=prev) ![After](kaloko:KALOKO-public/pricing)
```

KALOKO-public is the scenario id and pricing the step id. Without ?version= the reference shows the newest accepted screen; ?version=prev shows the one from the release before.

Draft the entry

```
npx kaloko changelog draft # prints the Unreleased entry
```

```
npx kaloko changelog draft --apply # writes it into CHANGELOG.md
```

The draft collects the pull requests and commits since the last release, groups them into Added, Changed, Fixed and Removed, and adds before and after pictures of the screens accepted runs show differently. Rewrite the lines for people: what they can do now, one change per line. --since <tag> drafts from another release.

Record the release

```
npx kaloko release 2.4.0
```

Unreleased becomes 2.4.0, the release is recorded with the accepted runs that are its pictures, and the command prints the changelog page. Without a version, Kaloko reads it from git tags, package.json, release-please or changesets.

When release-please, changesets or semantic-release make your releases, they keep writing CHANGELOG.md and Kaloko only reads it. npx kaloko ci github --release adds a workflow that records every published GitHub release.

Who reads it

Picture: open the docs on Kaloko to see it



The Products page · en · desktop

The changelog is under Products in the app. Everyone who sees one of the product's projects reads it, guests included. On Business and Enterprise an admin of those projects can choose on the page who else reads it:

- Readers from our domain: everyone who signs in from a verified domain of your organization.
- Anyone with the link: a public page with an Atom feed. Search engines index it only when you allow it.

Readers can also ask for a "What's new" e-mail on the changelog page (Business and up): each new release arrives with its pictures in their language.

How readers use the page: Read what's new. To keep guides current the same way, see Publish living docs.

Publish living docs and catch stale sections

Guides in your repository can show the screens of your accepted runs instead of pasted screenshots, so their pictures never go stale. When a screen changes after the text was written, Kaloko tells you which section to check. The pages you are reading now are made this way: kaloko.app/p/sinfin/kaloko/docs.

Set up the folder

```
npx kaloko init --docs=full
```

It creates docs/ with the four Diátaxis folders, each with a template:

| Folder | The reader wants | |---|---| | tutorials/ | to learn by doing, start to finish | | how-to/ | to get one task done |
| reference/ | to look a fact up | | explanation/ | to understand why |

index.md is the start page. Front matter is optional: title (else the first heading) and order (the position in the sidebar). Link pages with relative .md links.

Write a page

Pictures are step references, as in the changelog:

1. Choose **Sign in** in the top bar.
![The sign-in page](kaloko:KALOKO-public/login)

A translation sits next to its original: pay.cs.md beside pay.md, with the languages listed in the config (docs.languages: [en, cs], originals first). Pictures follow the language of the text.

One action per list item with its picture right under it also makes the page a walkthrough: readers choose Walk through it step by step and see one step at a time, and can tick I tried it.

Publish

```
npx kaloko docs publish # the docs of latest  
npx kaloko docs publish --version 2.4.0 # the docs of a release (Business)
```

Readers get a version selector, language and screen switches and search, and every picture opens its step on the canvas. kaloko release publishes the docs folder for its version on its own unless you pass --no-docs. Who reads the docs follows the same rules as the changelog: see Publish a changelog with pictures.

Catch stale sections

When you publish, Kaloko remembers the screen behind every reference. When a later accepted run shows a different screen, the section says "The screen changed in 2.4 — check the text." to the people who write the docs (Business).

```
npx kaloko docs check # against the published docs; needs a token  
npx kaloko docs check --offline # only the local files
```

Errors: broken references, steps the scenario files do not have, references without an accepted picture and stale sections. Warnings: missing or outdated translations and changelog items that people will see but that have no picture. The command exits with 1 on errors; --strict also on warnings, which suits CI.

For each stale section, open the step, compare it with the text and rewrite what no longer fits; the note clears at the next kaloko docs publish. If the text still fits, kaloko docs publish --reviewed clears the notes without a change. Use it only after you have read them.

To get the docs out as files for another site, see Export docs and the changelog.

Read what's new

Every release of a product has an entry in its changelog, written for the people who use it and illustrated with the screens it changed. The pictures come from runs someone accepted, so they show what was approved.

Find it

1. Open Products in the main menu and choose Changelog at the product, or open the link someone sent you. Each product has its own changelog; its newest release is on top.
2. The versions at the top jump to their release. Everything below a version is what changed in it, grouped as Added, Changed, Fixed and Removed.

Compare before and after

1. A change to a screen shows the screen before and after on top of each other. Drag the slider to move the split.
2. Side by side puts the two pictures next to each other (on a phone, one above the other).
3. Language and Screen switch every picture to another language or screen size, when the run captured them.
4. Open the flow opens the whole flow on the canvas, if you may see the run.

Keep up

- E-mail me what's new sends you the entry of every new release, with its pictures, in your language. The link at the bottom of each mail stops them.
- Download gives you the changelog as JSON, Markdown, offline HTML or PDF, for release notes or a client report.
- A public changelog also has a feed for feed readers.

Record a walkthrough video

A screenshot shows where a step ended. A recording shows how it got there: the spinner that never stopped, the menu that opened twice. Record when the screenshot cannot show what a reviewer has to judge: animations, loading states, drag and drop, a flaky step, or a demo for a client.

Record a walk

```
npx kaloko walk --record # each step, from the previous screen to its capture
npx kaloko walk --record flow # the whole walk per language, a chapter per step
npx kaloko walk --record failed # only steps whose script or checks fail
npx kaloko walk --record --record-quality low --steps checkout
```

To record by default, set it in the scenario or in `kaloko.config.yml`:

```
capture:
record: { mode: failed, quality: low, max_seconds: 45 }
nodes:
- id: checkout
record: true # this step always; record: false never
```

Quality is low, medium (the default) or high. Recordings are much larger than screenshots, so do not record every walk.

Traces come with every web step

Each walked web step keeps a trace: clicks and typing, network requests with status and timing, and console messages. It is small and on by default; `--no-trace` turns it off.

```
npx kaloko video list # what was recorded and traced, with sizes
npx kaloko video trace --step checkout # the step's trace as a timeline in the terminal
```

On the canvas the step detail shows Video next to the screenshot, and the Technical tab shows the trace. Click a row and the video jumps there. Replay plays the flow recording chapter by chapter.

Make one video of the run

```
npx kaloko video export --format mp4 # or webm, gif
npx kaloko video export --locale cs --steps cart,payment --lang cs --out checkout.mp4
```

The export puts the recordings in flow order with a title card before each step (its title and verdict). It is the file to send to a client or attach to a release note.

Tools you need

Web and Electron recordings need nothing but Chrome. With `ffmpeg` installed you also get MP4 for Safari, app recordings (Android, the iOS Simulator, macOS) and the walkthrough export. `npx kaloko video tools` says what this machine has. Walks in WebKit and Firefox are neither recorded nor traced.

Sharing

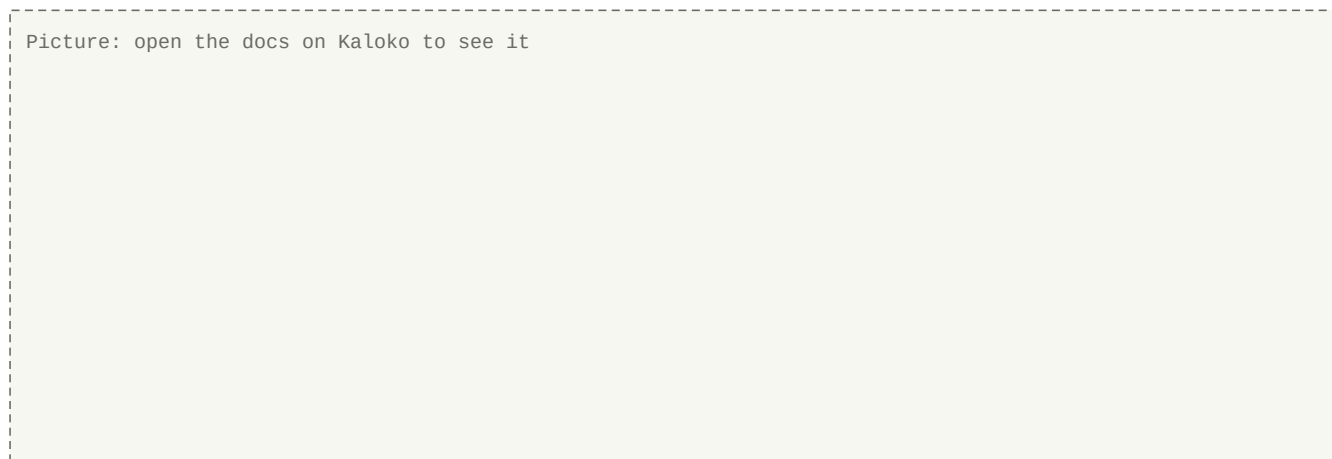
Recording works on every plan. `kaloko share` uploads recordings with the run on Business and Enterprise; on Free and Team they stay in the local preview, and the screenshots and traces are shared. Shared recordings are kept for 30 days (Enterprise sets its own period in Settings -> Storage). A recording over 25 MB stays local: lower the quality or `max_seconds`.

Review a run

A run is one pass of an agent through a flow: every screen it reached, in every language and screen size it walked, with the checks it made. You review it in the browser; there is nothing to install.

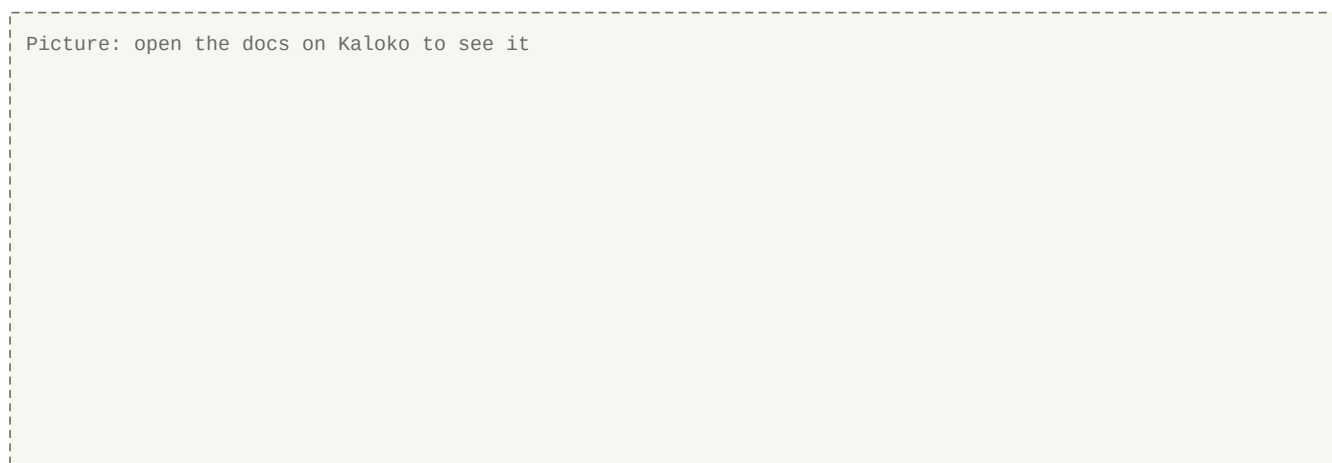
Open the run

1. Open the link from the pull request, the issue or the e-mail. The run opens on the canvas: every step of the flow as a map, with arrows for the way between them.



The map of a run · en · desktop

2. If you would rather work through a list, press I. The same run becomes a list of steps with their results, and you can search it and filter by status.

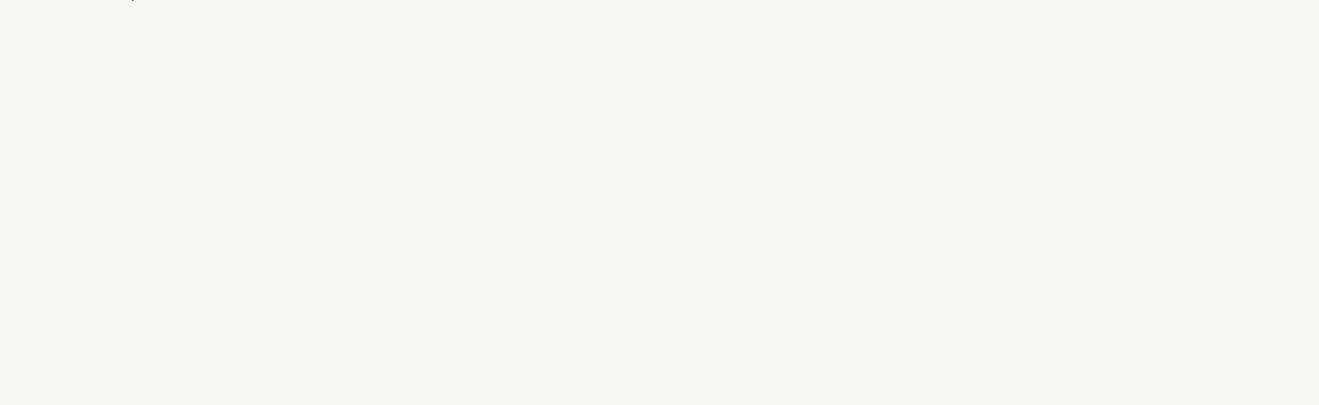


The run as a list · en · desktop

Look at a step

1. Click a step. Its detail shows the screenshot, the criteria with what was found, and the comments people left on it. <- and -> move to the previous and next step.

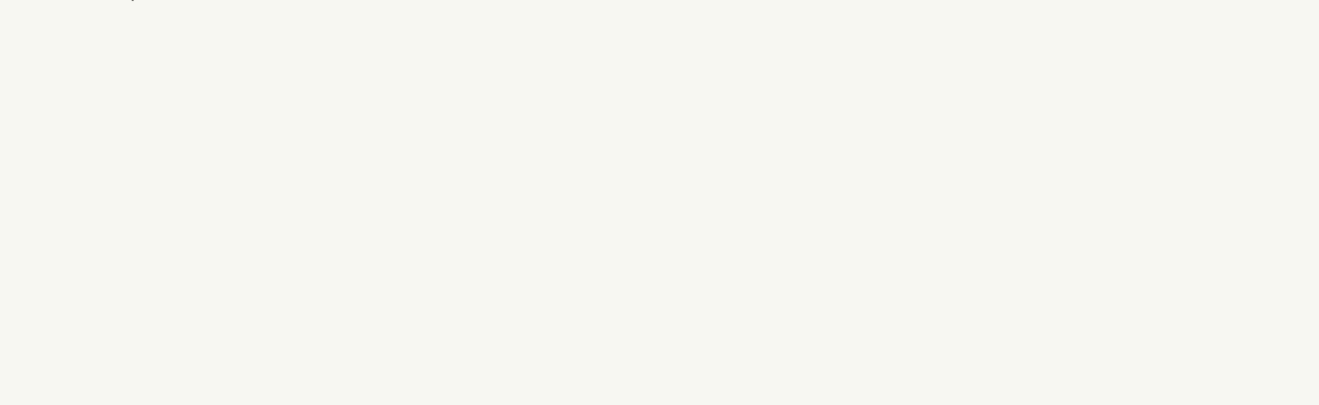
Picture: open the docs on Kaloko to see it



A step with its comments · en · desktop

2. Something looks wrong? Write a comment on the step. All open comments of the run sit in the comments panel, so nothing gets lost between steps.

Picture: open the docs on Kaloko to see it

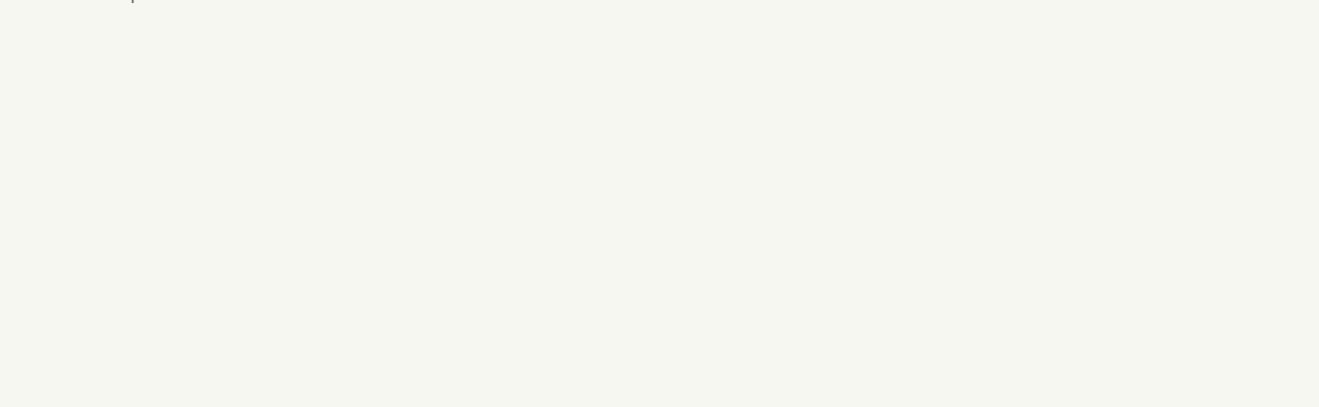


The comments panel · en · desktop

Walk it like a user

Press R to replay the run: one screen at a time, in the order a person would see them. It is the quickest way to show the flow to someone who has never seen the canvas.

Picture: open the docs on Kaloko to see it



Replay · en · desktop

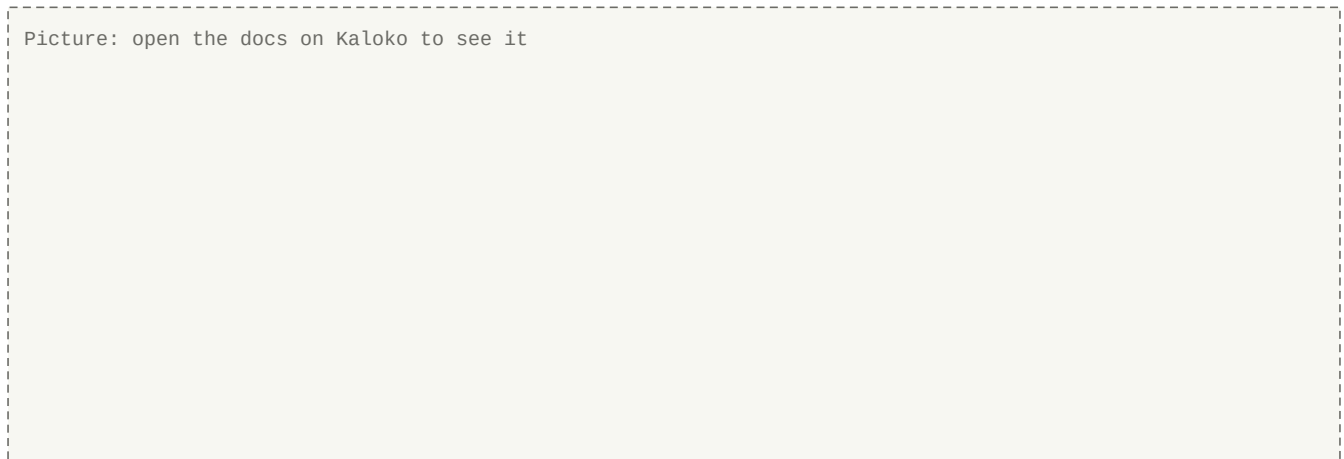
When you know what you think, approve or return the run.

Review only what changed

Review mode shows one step at a time, starting with the steps that need a person: failed and returned steps, criteria the AI was unsure about, steps that changed since the last accepted run and steps with open comments. Steps that did not change keep the approval someone gave them before.

Open review mode

1. Open the review link from the pull request or the e-mail (it ends in #review), or open the run and press Q.



Review mode · en · desktop

2. The step you are deciding sits next to the accepted run, so you see what moved.

Decide with the keyboard

| Key | What it does | |---|---| | J / K | next and previous step | | A | approve the step | | R | return the step with a note | | C | comment | | G | the step in every language and screen size |

When every step is decided, accept or return the whole run.

Look closer

In the large view you can compare the two runs side by side, as a difference, with Swipe (drag a divider across both screenshots) or with Onion skin (this run faded over the other). To point at a problem, draw a box, an arrow or a line on the screenshot while you write the comment, or paste a picture.

Approvals carried over

A carried approval names who gave it and in which run. If you think the step needs another look, revoke it and the step waits again. A carried approval never accepts the run for you.

Roles and API tokens

Every person in an organization has a role, and every token acts for a person with no more rights than that role. Agents and CI use tokens; they never take a seat.

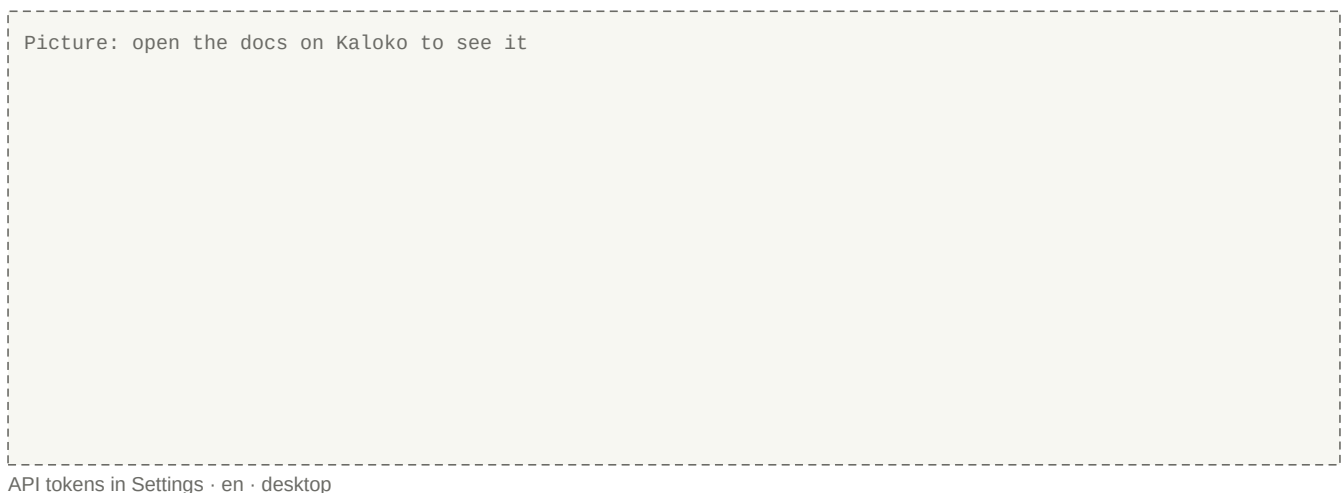
Roles

| Role | May do | Seat | |---|---|---| | Viewer | read runs and designs, comment, approve and return | free | | Designer | also work on Kaloko Design drafts, versions, branches and token sets | one seat | | Developer / Tester | also upload and manage runs and their own tokens | one seat | | Admin | also manage members, domains, all tokens and the organization's settings | one seat |

A project can have its own project admin, who invites people to that project, changes their roles and can restrict or open it. Developers / testers and admins can create projects; whoever creates one becomes its admin. Change a member's role in Settings -> People.

Create a token in the app

1. Open Settings -> API tokens.



2. Name the token after where it lives, for example "laptop CLI" or "GitHub Actions".

3. Tick what it may do. Reading is always included.

- comment and approve: comments and verdicts, as you
- design: kaloko sync of design drafts
- upload runs: kaloko share from the CLI or CI
- everything my role allows

4. Pick how long it lives: 30 days, 90 days or a year.

5. Optionally limit it to chosen projects; it then sees only those and cannot create new ones.

6. Copy the token once and put it into the project's .env as KALOKO_TOKEN.

The list shows each token's owner, role, prefix, last use and expiry. Revoke ends a token at once.

Sign in from the terminal

On your own machine there is a shorter way:

```
npx kaloko login
```

The browser opens; you approve the code, choose the organization and what the token may do. The token is valid for at most 30 days and is saved to ~/.config/kaloko/.env. --print prints it instead, --no-open shows the link

without opening a browser.

Tokens for CI

Create a token with the upload runs scope, ideally limited to the project, and store it as a KALOKO_TOKEN secret in your CI. npx kaloko ci github and ci gitlab write workflows that read it; see the CLI reference.

Rules an admin can set

Under Settings -> Security an admin can allow only admins to create tokens with every right, turn off tokens limited to projects, and decide whether guests may create tokens at all. A guest's token sees only their projects. On Enterprise, service accounts hold tokens for automation that belongs to no person.

Run Kaloko in CI and comment on pull requests

In CI, Kaloko runs the same commands as on your laptop. The usual setup walks what a pull request changed on its preview deployment (Vercel, Netlify, Cloudflare Pages or any review app) and puts the canvas link into a comment on the pull request.

1. Write the workflow

```
npx kaloko ci github # writes .github/workflows/kaloko.yml
npx kaloko ci gitlab # writes kaloko.gitlab-ci.yml; include it from .gitlab-ci.yml
npx kaloko ci github --print # show it, write nothing
```

--scenario a.yml,b.yml limits it to some scenarios. The command also adds a preview environment to kaloko.config.yml with base_url: \${KALOKO_PREVIEW_URL}. A scenario that lists environments: has to include preview, or kaloko start refuses it; the command tells you which ones.

2. Add the token

Create a token with the tester role in Settings -> API tokens and add it to the repository's CI secrets as KALOKO_TOKEN. On GitLab, add KALOKO_GITLAB_TOKEN too: a project access token with the api scope, so Kaloko can write the note on the merge request. Commit the workflow.

What the workflow does

On every successful preview deployment of a pull request:

1. It installs the project and the browser (npx kaloko browser install --with-deps), plus WebKit and Firefox when a scenario asks for them.
2. npx kaloko ci preview-url --wait 600 --github-env finds the preview address, the pull request and its base branch and passes them on as KALOKO_PREVIEW_URL, KALOKO_PR and KALOKO_BASE.
3. For each scenario, kaloko affected --check skips it when the change does not reach any of its steps. Otherwise it runs:

```
npx kaloko start --scenario qa/flows/checkout.yml --env preview
npx kaloko walk --ephemeral --affected
npx kaloko share --pr
```

share --pr adds one comment per scenario to the pull request and updates it on later pushes. Uploads go by content, so screenshots that did not change are not sent again.

Good to know

- walk --affected needs the git history: check out with fetch-depth: 0 on GitHub or GIT_DEPTH: "0" on GitLab.
- --trigger pull_request runs on every push instead of the deployment event and waits for the preview.
- kaloko ci preview-url prints only the address, so it works in any CI: export KALOKO_PREVIEW_URL="\$(npx kaloko ci preview-url)".
- A preview behind Vercel's deployment protection needs a VERCEL_AUTOMATION_BYPASS_SECRET secret and the access.headers block of the preview environment, which the generated config has commented out.
- No comment on the pull request? Usually there was no preview deployment, or the token is missing. The job log says which.

When release-please, changesets or semantic-release publishes your releases, npx kaloko ci github --release adds a second workflow that records each release on Kaloko. All options are in the CLI reference.

Walk Android, iOS and desktop apps

Kaloko walks native apps the way it walks web pages: every step becomes a screenshot with its UI tree, checked criteria and a place on the canvas. The scenario says which platform it is, and the config says which app to open.

1. Name the app per environment

```
# kaloko.config.yml
environments:
staging:
app:
android: { package: com.acme.shop.staging, apk: build/app-staging.apk, reset: true }
ios: { bundle: com.acme.shop, app: build/Shop.app, device: iPhone 17 }
desktop:
app:
electron: { main: dist/main.js }
# desktop: { name: Notes } # a macOS app by its window
# windows: { name: contoso, path: dist/Contoso.exe }
```

2. Set the platform in the scenario

```
platform: android # android | ios | electron | desktop | windows
devices: [pixel-8-api-34, pixel-7-api-33]
color_schemes: [light, dark]
nodes:
- id: product
kind: screen
path: shop://product/42 # a deep link
packs: [app]
```

Steps are reached by deep links or by a step script (`app.tap(...)`, `app.type(...)`). Every locale starts the app fresh in that language, and `reset: true` clears its data first.

What each platform can do

| Platform | Driven by | Notes | |---|---|---| | Android | adb: emulator, USB phone or a phone over Wi-Fi | an emulator that is not running is started; dark mode and reduced motion are set per capture | | iOS Simulator | xcrun simctl | taps and the UI tree need idb; without it Kaloko reads the text from the screen | | iPhone and iPad | devicectl and a screenshot tool | capture only, text read from the screen | | Electron | Playwright | the window is a web page: every web check applies | | macOS app | the app's window and accessibility tree | capture only; your tool, AppleScript or a person moves between screens | | Windows app | UI Automation through PowerShell | taps and typing; walk it on Windows |

The app pack checks labels on controls, touch target sizes, cut-off text and crashes.

Screens from other tools

Maestro, Appium, XCUITest, Detox or a person can take the screenshot, and Kaloko lays it out with its UI dump:

```
npx kaloko capture --step checkout --image shot.png --tree hierarchy.xml --device "Pixel 8"
--density 2.625
```

--tree reads uiautomator or Appium XML, idb ui describe-all --json and Maestro's hierarchy JSON. Without a tree, --ocr reads the text from the image on macOS. A whole folder of screenshots or a test report goes in with `npx kaloko import`.

Check the machine first

`npx kaloko doctor`

It says which app tools are missing on this machine (adb, simctl, idb, ffmpeg) and how to install them. On macOS, kaloko doctor --fix asks for the Screen Recording and Accessibility permissions desktop captures need.

Write a scenario

A scenario is the flow Kaloko walks: a YAML file with the screens, the way between them and what each screen has to meet. Task flows usually sit next to the task (docs/tasks/TASK-123/qa/scenario.yml), long-lived flows such as the checkout in qa/flows/.

Let the agent write it

With the skill installed, ask your agent instead of typing YAML:

Read the task in docs/tasks/TASK-123, write its acceptance plan and a Kaloko scenario, and validate it.

The agent writes two files: ACCEPTANCE.md for people (goal, steps, criteria AC1, AC2...) and the scenario for the runner. When the intent changes, it updates the plan first and then the scenario.

The parts of the file

```
schema: qawalk.scenario.v1
id: TASK-123-checkout
title: Checkout
kind: task # task | flow
product: shop # optional, with module:
locales: [en, cs]
viewports:
desktop: [1280, 800]
mobile: [390, 844]
environments: [local, staging]
nodes:
- id: cart
kind: screen
title: Cart
path: /cart
purpose: The visitor sees what they are buying and can go on to pay.
execution:
strategy: playwright # playwright | agent | manual
script: steps/cart.mjs
criteria:
- id: AC1
text: The page has exactly one main heading.
check: { type: deterministic, assert: count, selector: h1, min: 1, max: 1 }
edges:
- { from: cart, to: payment, label: continue }
```

- Step ids (cart) stay stable: links, comments, approvals and docs pictures refer to them.
- purpose is one sentence about what the screen is for. The evaluator reads it.
- execution says who moves to the screen: a Playwright script, an agent that captures with kaloko capture, or a person.
- Other node kinds: decision for a branch, email for a message from the mailbox, stack for many pages of one template.

Every key is in the scenario file reference. To add checks, see Add criteria and check packs.

Check it

```
npx kaloko validate qa/flows/checkout.yml
npx kaloko validate qa/flows/checkout.yml --coverage
```

validate checks the schema and the references between steps. --coverage compares the criteria with the

acceptance plan and lists what the plan asks for and the scenario does not check.

Start from public pages

For a marketing site or documentation, start from a draft:

```
npx kaloko draft https://example.com/ https://example.com/pricing
```

Kaloko fetches the pages and writes a read-only production scenario and its plan, with the page title, H1 and description as the purpose and the seo, a11y, perf and console packs. It also adds a step for an address that does not exist, so the 404 page is checked too. Then sharpen each purpose and add what the page really has to meet.

Criteria, checks and packs

A criterion is one thing a step must meet. It has an id (AC1), a text for people and exactly one check. How to add them to a scenario: Add criteria and packs. How the results turn into a verdict: How verdicts are decided.

Kinds of check

| type | When to use it | Options | |---|---|---| | deterministic | anything countable or exact: presence, counts, text, the URL, styles, values in the page | assert and its options (below) | | semantic | judgement: is the copy clear, is there one primary action | question (English, yes = met), true and false (what counts as yes and no), scope (CSS selector sent to the evaluator), about: console (read the console errors instead of the page), vision: true (also show the screenshot) | | manual | neither works; a reviewer decides on the canvas | note |

Numbers, dates and counts belong in deterministic checks, never in a semantic question. Ask one thing per question.

Deterministic checks (assert)

Common options: selector (CSS), pattern (a regex) with flags (i, m, s), value, min, max, all (every match must pass, not just one).

Page content

| assert | Passes when | |---|---| | exists | selector matches at least one element | | absent | selector matches nothing | | count | the number of matches is between min and max | | text_matches | the element's visible text matches pattern | | text_absent | no visible text matches pattern | | text_equals | the whole text equals value (whitespace collapsed) | | attr_equals | attribute attr equals value | | attr_matches | attribute attr matches pattern | | no_overflow | the page does not scroll sideways | | no_console_errors | the console has no errors |

Address, metadata and links

| assert | Passes when | |---|---| | title_matches | the page title matches pattern | | meta_matches | the content of the selector meta tag matches pattern | | link_matches | the href of the selector link matches pattern | | url_matches | the URL after the step matches pattern; part picks href, path, query, hash, host or origin | | url_equals | the URL equals value (a path or a whole URL); ignore_query drops ?... and #... | | http_status | the document's status is between min and max (default 200–299), or equals value | | link_home | the page links back to the home page | | links_ok | every same-origin link answers below 400 (max links per page, pattern skips paths, follow outside read-only environments) | | hreflang_pairs | at least min languages, each version answers 200 and links back |

Styles and values in the page

| assert | Passes when | |---|---| | style_equals | the computed CSS property equals value or a design token; op compares numbers ($\geq$), tolerance allows px rounding | | style_matches | the computed property matches pattern | | js_equals | one read-only JavaScript expression in the page returns value (or matches pattern, or compares with op) |

Accessibility (the a11y pack uses these)

| assert | Passes when | |---|---| | a11y_images_alt | images have alternative text | | a11y_names | links and buttons have an accessible name | | a11y_labels | form fields have labels | | a11y_headings | heading levels are not skipped | | a11y_contrast | text meets WCAG AA contrast | | a11y_main | the page has a main landmark | | a11y_unique_ids | element ids are unique | | a11y_focus_order | Tab order follows the page and reaches every control | | a11y_focus_visible | focus is visible with at least 3 : 1 contrast | | a11y_keyboard_trap | Tab never gets stuck | | a11y_skip_link | a skip link or the main content within three Tab stops | | a11y_ax_names | controls have a name in the browser's accessibility tree | | a11y_landmarks | landmarks are complete and repeated ones labelled | | a11y_aria | ARIA roles and attributes are valid |

Performance, apps and design

| assert | Passes when | |---|---| | perf_lcp, perf_cls, perf_ttfb, perf_weight | largest contentful paint, layout shift,

time to first byte or page weight stays within max | | app_labels | every app control has an accessible label | | app_touch_targets | touch targets are at least 48 dp (Android) or 44 pt (iOS) | | app_no_truncation | no text is cut off with an ellipsis | | app_no_crash | the app did not crash or stop responding | | design_pixels | the screen differs from the design reference in at most ratio of pixels (default 0.03) | | design_structure | headings, landmarks, actions and fields match the design reference | | uses_tokens | colours, fonts, sizes, spacing and radii come from the token set |

Packs

packs: [seo, a11y] on a step adds a fixed set of criteria. A criterion of your own with the same id replaces the pack's, which is how you change a limit (AC401 with max: 4000).

| Pack | Ids | What it adds | |---|---|---| | seo | AC101–AC111 | one H1, title 10–70 and description 50–160 characters, canonical, hreflang for every language, Open Graph, not noindex, viewport meta, html lang, no overflow, no console errors | | usability | AC201–AC204 | the purpose is clear on the first screen, one primary action, labelled fields, link texts that say what they do | | a11y | AC301–AC314 | every accessibility check above; the walk also presses Tab through the page, without clicking or typing | | perf | AC401–AC404 | LCP within 2.5 s, CLS below 0.1, TTFB within 800 ms, page weight within 3 MB (measured in the walk's browser) | | console | AC501 | the evaluator reads the console errors and says whether they touch what the step accepts | | errors | AC601–AC606 | a real 4xx/5xx status, a link home, navigation or search, no stack trace, a plain message with a way forward, no console errors | | app | AC701–AC705 | labels, touch targets, no truncated text, no crash, a clear purpose and next action | | design | AC801–AC803 | the implementation against the design reference: pixels, structure, tokens |

The a11y pack covers part of WCAG 2.2 AA. It does not replace an audit with a screen reader.

CLI commands

Every command of the kaloko CLI with what it does. Run them with `npx kaloko <command>` in a project that has the package installed. `npx kaloko help` prints the same list with every option.

Options that work everywhere: `--run <id|dir>` (or `KALOKO_RUN`) picks the run, `--config <file>` the config file, and `KALOKO_AGENT=<name>` labels what an agent writes.

Set up a project

| Command | What it does | |---|---| | `kaloko init` | Creates `kaloko.config.yml`, installs the skill for Claude Code or Codex (`--agent`), detects the framework, dev server, languages and main routes, drafts a first scenario; `--docs=off|changelog|full` sets up the changelog and the docs folders | | `kaloko login` | Signs this machine in through the browser and saves a token (at most 30 days) to `~/.config/kaloko/.env`; `--print` prints it instead | | `kaloko doctor` | Checks Node, Chrome, the config, environments, browser ports, the token, the evaluator key and app tools; `--fix` asks macOS for capture permissions | | `kaloko draft <url...>` | Drafts a first plan and a read-only scenario for public pages, with the seo, a11y, perf and console packs | | `kaloko validate <file...>` | Checks scenarios against the schema and their references; `--coverage` compares the criteria with the acceptance plan | | `kaloko scenarios` | Lists the scenarios the config matches | | `kaloko ci github|gitlab` | Writes a workflow that walks what each pull request affects on its preview deployment and comments on it | | `kaloko ci github --release` | Writes a workflow that records every release your release tool publishes | | `kaloko ci preview-url` | In CI: finds the preview URL, the pull request and its base |

Walk and capture

| Command | What it does | |---|---| | `kaloko start` | Creates a run of `--scenario` on `--env` and makes it current; `--locales`, `--viewports`, `--browsers`, `--color-schemes`, `--devices` and more choose what to capture | | `kaloko walk` | Runs the Playwright steps and captures them, then evaluates; `--affected`, `--record`, `--heal`, `--stability N`, `--concurrency N`, `--ephemeral` | | `kaloko capture --step <id>` | Captures the current page for a step; `--crop` attaches an element, `--image` imports a screen taken by another tool | | `kaloko open --step <id>` | Opens the step's URL in the shared browser | | `kaloko mail --step <id>` | Captures an e-mail from the environment's mailbox, or a saved message with `--file` | | `kaloko browser start|stop|status` | One shared browser per environment that agents, scripts and people drive | | `kaloko browser install` | Downloads the browser engines (`--browsers chromium,webkit,firefox`) | | `kaloko devices` | Lists the device presets you can use as viewports | | `kaloko session reset` | Starts as a new visitor: clears cookies and storage in the shared browser | | `kaloko auth save` | A person signs in once (SSO, MFA) and the walk reuses the session; `kaloko auth list` shows saved sessions | | `kaloko affected` | Says which steps the changed files reach, and why | | `kaloko import <path>` | Lays out screenshots, Playwright reports, traces or a walkthrough as a run | | `kaloko status` | Shows what the current run has captured |

Results and review

| Command | What it does | |---|---| | `kaloko evaluate` | Runs the deterministic checks and the semantic evaluator for all captures; `--recheck` after a scenario change | | `kaloko build` | Writes `run.json` and the local preview | | `kaloko preview` | Builds the preview and prints its path | | `kaloko share` | Uploads the run to `kaloko.app`; `--pr` comments on the pull request or merge request | | `kaloko review` | What a person still has to decide on a shared run, and the review-mode link | | `kaloko feedback` | Open comments from reviewers, returned steps first | | `kaloko comment` | Comments, replies, resolves and reopens feedback from the terminal | | `kaloko compare` | A text diff of two runs: statuses, verdicts, scores and evidence | | `kaloko stability` | Finds regions that change on every run; `--apply` adds them to the scenario, `--pull` brings in regions drawn on `kaloko.app` | | `kaloko calibrate` | Compares the semantic evaluator with human labels and suggests thresholds | | `kaloko signoff <run-id>` | Downloads the signed acceptance record of an accepted run; `--verify <file>` checks one without a token | | `kaloko video export` | Turns the run's recordings into one video; `kaloko video list`, `trace` and `tools` show recordings, traces and tools | | `kaloko screens` | Searches the newest screenshot of every step across your flows | | `kaloko inbox` | Your notifications on `kaloko.app` |

Shared runs

| Command | What it does | |---|---| | kaloko projects | Projects of the organization | | kaloko runs | Shared runs, the newest per scenario; --all lists every run | | kaloko trash <run-id...> | Moves runs to the trash; kaloko restore <run-id> brings one back | | kaloko prune | Keeps the newest runs per scenario and environment and lists the rest for the trash; --yes moves them | | kaloko keep <run-id> | A kept run never expires and is never pruned; --off releases it | | kaloko archive <scenario-id> | Hides a retired scenario from the list; its runs stay | | kaloko public | Makes a shared run readable without signing in | | kaloko export | Downloads a shared run as a zip; --scenario the newest capture of every step; --format the docs and changelog |

Jira, Linear and hosted runs

| Command | What it does | |---|---| | kaloko issue create | An issue in Jira or Linear from a returned step or a failing criterion; kaloko issue list shows them | | kaloko schedule add | Kaloko opens the pages of a read-only scenario every day or week and shares the run | | kaloko schedule list\run\pause\resume\remove | Manages the schedules | | kaloko schedule export github | A GitHub Actions workflow that walks the scenario on a schedule with the full CLI |

Products, changelog and docs

| Command | What it does | |---|---| | kaloko products sync | Puts the products of the config on Kaloko without recording a release | | kaloko products list | The products you read on Kaloko, and those of the config that are not there yet | | kaloko changelog draft | Writes the Unreleased entry from merged pull requests and changed screens; --apply writes it into CHANGELOG.md | | kaloko release [<version>] | Records a release with its changelog entry, pictures and docs | | kaloko docs publish | Sends the docs folder to Kaloko as the docs of a version | | kaloko docs check | Broken references, stale sections, missing translations and changelog items without a picture; exits 1 on errors | | kaloko export --format <f> | The docs and changelog as Markdown, Docusaurus, VitePress, MkDocs, HTML, PDF or JSON |

Design

| Command | What it does | |---|---| | kaloko sync | Exchanges design revisions with the shared draft (--pull, --push) | | kaloko design lint | Checks the design folder against the portable HTML rules | | kaloko design import figma | Turns Figma frames into design steps and variables into tokens | | kaloko history | Versions of the flow and revisions of a step | | kaloko spec [step] | What to build for an approved design step: elements, styles mapped to tokens, states | | kaloko drift [step] | How the implementation differs from the approved design; exits 1 while it differs | | kaloko reference | Shows or sets the design reference and the baseline | | kaloko tokens pull\push\check\validate | The project's design tokens (DTCG) | | kaloko branch list\create\merge | Branches of the design draft |

kaloko help prints all of this with every option.

Configuration file

kaloko.config.yml sits in the root of the repository and tells the CLI where your flows run. kaloko init writes a commented one; kaloko doctor checks it. Its format is qawalk.config.v1, and schema, org and environments are required.

```
schema: qawalk.config.v1
org: acme
project: shop
scenarios: [qa/flows/*.yml]
environments:
  local:
    base_url: http://localhost:3000
  staging:
    base_url: https://staging.acme.test
  access:
    basic: { user_env: STAGING_USER, password_env: STAGING_PASSWORD }
  accounts:
    member: { user: qa@acme.test, password_env: MEMBER_PASSWORD }
  production:
    base_url: https://acme.com
    readonly: true
    blocked_paths: ["/account/**"]
  walk:
    browsers: [chromium]
  share:
    expires_days: 30
```

Top-level keys

| Key | What it holds | |---|---| | schema | always qawalk.config.v1 | | org | the organization's slug on kaloko.app | | project | the Kaloko project for runs; default: the repository's name | | scenarios | glob patterns of scenario files | | output | the run folder (default tmp/kaloko) | | environments | where flows run (below) | | browser | channel (a Playwright channel such as chrome; empty = the bundled Chromium) and port of the shared browser (default 9333) | | walk | defaults for every run: concurrency (languages at once, 1–16), browsers, color_schemes, reduced_motion, forced_colors, display_modes | | capture | defaults for every scenario: record, trace | | evaluators | the semantic evaluator jev and the vision evaluator (below) | | share | url (default https://kaloko.app), token_env (default KALOKO_TOKEN), expires_days (default 30) | | docs | changelog and docs settings (below) | | products | what you release and version as a whole (below) |

Environments

Each key under environments is a name you use with kaloko start --env.

| Key | What it holds | |---|---| | kind | app (default) or design (a folder of live HTML steps) | | base_url | the address, or one per language with a default fallback; \${VAR} is filled from the environment (review apps) | | readonly | true for production: the CLI only reads, never signs in to back offices, never creates data | | blocked_paths | path globs a read-only run never opens | | allowed_post_paths | path globs a read-only run may still POST to (a cart) | | may_create_data | false forbids creating data (default true) | | access | protection in front of the whole environment: basic (user_env, password_env), headers (service tokens), client_certificate, origins that receive them | | accounts | people who sign in, keyed like the scenario's account: user or user_env, password_env, totp_env, storage_state (a sign-in saved by kaloko auth save) | | mail | the mailbox: provider (mailpit, script, none), url, user_env, password_env, address, script, options, auth | | app | the app under test per platform: android, ios, electron, desktop, windows | | serve, fixtures, tokens_mode | design environments: the folder of steps, JSON fixtures, the token mode | | html_view | experimental: keep a static HTML view of every capture |

Evaluators

| Key | What it holds | |---|---| | evaluators.jev.api_key_env | the variable with your own evaluator key (default TYPESAFE_API_KEY) | | evaluators.jev.hosted | without a local key, evaluate through your plan's allowance (default true); false uses only your key | | evaluators.jev.thresholds | pass and fail scores | | evaluators.jev.model, base_url, max_state_chars, signals, concurrency | finer settings, rarely needed | | evaluators.vision | the second evaluator that looks at the screenshot (provider anthropic, key in ANTHROPIC_API_KEY); false turns it off |

Docs and products

| Key | What it holds | |---|---| | docs.dir | docs sources (default docs) | | docs.changelog | the changelog file (default CHANGELOG.md) | | docs.style | keep-a-changelog or release-notes | | docs.version_source | auto, git-tag, package.json, release-please, changesets, semantic-release, pyproject, cargo, gemspec, manual | | docs.languages | languages of the docs; the first is the originals' | | docs.framework | auto, docusaurus, vitepress, mkdocs, next, plain | | docs.publish | who reads: members, domain or public (the last two need an admin) | | docs.images | viewport and locale (reader or a language) of the pictures | | products.<slug> | name (text or per language), version (source, path, pattern), docs, project or projects, modules, reference_env, default_scenario, publish |

Without products, the project is the product.

Secrets

Secrets never go into the config. Wherever one is needed, the config names an environment variable (password_env: STAGING_PASSWORD, { env: CF_ACCESS_CLIENT_SECRET }). Kaloko reads the value from:

1. the environment of the process,
2. .env next to kaloko.config.yml,
3. ~/.config/kaloko/.env.

A variable already set wins over the files. Values are removed from captured HTML, page text and console output. kaloko doctor says which named variables are missing. The token for sharing is KALOKO_TOKEN unless share.token_env says otherwise; see Roles and tokens.

MCP tools

Kaloko's MCP server lets an agent read runs, feedback, designs, the changelog and docs, and answer reviewers. It is part of the Team plan and up. Capturing, evaluating and uploading stay in the CLI.

Connect

The server is <https://kaloko.app/mcp> (MCP over Streamable HTTP). Authenticate with an organization API token in the Authorization header; how to create one: Roles and tokens.

```
claude mcp add --transport http kaloko https://kaloko.app/mcp --header "Authorization: Bearer $KALOKO_TOKEN"
```

Other agents (Codex, Cursor) take the same URL and header as an HTTP MCP server. A read-only token is enough to read; tools that write need a token with that scope, as noted below. Screenshots and files are not sent over MCP: the answers carry their URLs, which the same token can fetch.

Runs and review

| Tool | What it does | |---|---| | `get_started` | where the organization stands, its sample run and the next steps | | `add_sample_run` | adds the sample run, or returns the one there is | | `list_projects` | projects of the organization with their slugs | | `list_runs` | shared runs, newest first, filtered by scenario, project, verdict, author, environment or branch | | `get_run_summary` | one run: statuses per step and language, criteria that need attention with evidence and reasons, screenshot URLs, the canvas link | | `get_stack_items` | the pages of a stack step with their status and failing criteria | | `get_step_assets` | the newest capture of every step of a scenario at stable URLs (Business) | | `get_step_recording` | a step's video and trace | | `get_criteria_verdicts` | reviewers' verdicts on manual or overridden criteria | | `get_review_queue` | what a person still has to decide in a run, and the review link to pass on | | `decide_step` | the agent's own approve or return on a step; marked as an agent's, never counted as a person's | | `revoke_carried_approval` | takes back an approval carried over to a step that should be looked at again | | `get_signoff_record` | the signed acceptance record of an accepted run (Business) | | `get_affected_steps` | which steps of a scenario the changed files reach, and why | | `list_notifications` | the token owner's inbox |

Feedback and issues

| Tool | What it does | |---|---| | `get_feedback` | the run's verdict, approved steps, open returns and comments | | `list_comments` | open threads across the organization, with replies, pins and drawn regions | | `add_comment` | a comment on a step, a run or a whole flow, or a reply | | `resolve_comment` | closes a thread with a note on what changed | | `reopen_comment`, `edit_comment`, `delete_comment` | reopen a thread; edit or delete your own comment | | `list_ignore_regions` | regions people drew to leave out of pixel diffs | | `add_ignore_region` | suggests such a region | | `create_issue` | a Jira or Linear issue from a step, with the screenshot, criterion and notes | | `list_issues` | issues created from steps, with their state | | `list_schedules`, `run_schedule` | hosted runs and their page budget; start one now |

Design

| Tool | What it does | |---|---| | `get_draft` | which revision each step of the shared draft shows, and who changed it | | `get_step` | the handoff of a step: live files, tokens, page structure, open comments | | `get_step_spec` | what to build for a step as approved: elements, styles mapped to tokens, states, contrast | | `get_step_drift` | how an implementation run differs from the design reference | | `get_step_history` | versions and revisions of a step with their authors | | `list_step_changes` | steps that changed between two references or since the previous accepted run | | `get_references`, `get_tokens` | the design reference and baseline; the project's token set | | `get_draft_feedback` | comments on the draft | | `get_version_notes`, `set_version_notes` | read or rewrite what changed in a version | | `publish_version`, `set_reference`, `set_step_status`, `add_draft_comment` | publish the draft as a version, set the reference, set a step's status, comment on the draft |

Screens, changelog, docs and products

| Tool | What it does | |---|---| | search_screens | the screen library: the newest screenshot of every step, searched by words, project, product, language or status | | list_products | the products you read, with their newest release, docs and readers | | get_changelog | a product's visual changelog, optionally with the pictures resolved | | draft_changelog_entry | what accepted runs since the last release show differently, as a draft entry with before and after references | | resolve_step_image | one step reference (kaloko:<scenario>/<step>) as an image URL | | get_docs | a product's published docs: versions, languages, the sidebar, or one page | | docs_check | what needs attention in the published docs: broken references, stale sections, missing translations |

Errors come back as tool results marked as errors (an unknown run, a missing argument). A token that is not valid gets HTTP 401; an organization below Team is told the MCP server is not in its plan.

Scenario file

A scenario is a YAML file (usually in qa/flows/) that describes one flow: its steps, the arrows between them and what each step must meet. Its format is qawalk.scenario.v1; kaloko validate <file> checks a file against it. How to write one step by step: Write a scenario.

```
schema: qawalk.scenario.v1
id: WEB-12-checkout
title: Checkout
locales: [en, cs]
viewports: { desktop: [1280, 800], mobile: iphone-15 }
environments: [local, staging]
nodes:
- id: cart
  kind: screen
  title: { en: Cart, cs: Košík }
  path: /cart
  purpose: The shopper sees what they are buying and can continue.
  execution: { strategy: playwright, script: steps/cart.mjs, instructions: Add one product and
    open the cart. }
  criteria:
  - { id: AC1, text: Shows the total, check: { type: deterministic, assert: exists, selector:
    .cart-total } }
  - id: payment
  kind: screen
  title: Payment
  path: /checkout/payment
  packs: [a11y]
  edges:
  - { from: cart, to: payment }
```

Top-level keys

schema, id, title, nodes and edges are required.

| Key | What it holds | |---|---| | schema | always qawalk.scenario.v1 | | id | the scenario's id (lowercase letters, digits, dashes); a ticket in it (WEB-12-...) links the issue | | title | a name for people | | kind | task (default) or flow | | archived | true hides a retired scenario from the list; its runs stay reachable | | source | path to the acceptance plan (ACCEPTANCE.md), used by kaloko validate --coverage | | version | a number you raise when the flow changes | | locales | languages to walk, e.g. [en, cs] | | viewports | names with a size ([1280, 800]), a device preset (iphone-15, ipad-landscape), or a list of preset names | | browsers | chromium (default), webkit, firefox | | color_schemes | light, dark | | reduced_motion | no-preference, reduce | | forced_colors | none, active (Windows high contrast) | | display_modes | browser, standalone (an installed web app) | | devices | app scenarios: the emulators, simulators or phones to walk on | | environments | environments of the config the scenario may run on | | readonly | true for a scenario that only reads; required on a read-only environment | | platform | web (default), android, ios, electron, desktop, windows | | tags | free labels | | nodes | the steps (below) | | edges | arrows between steps (below) | | lanes | labels of the rows on the canvas, index = lane number | | issue | the ticket: number, repo (owner/name), key (Jira or Linear, ENG-42), url, title | | sessions | people in the flow, each with title, optional mailbox, account, blocked_on | | project | the Kaloko project for its runs; overrides the config | | product, module | which product and module of products: the flow shows, for the changelog and docs | | capture | capture quality for every step (below) |

Step keys (nodes)

Every step needs id, kind and title.

| Key | What it holds | |---|---| | id | the step id, used in links, --steps and step references (kaloko:<scenario>/<step>) | | kind | screen, email, external (a third-party screen), decision, stack (many pages

of one template) | | title | text, or one per language ({ en: Cart, cs: Košík }) | | path | the address relative to base_url, or one per language; a deep link in app scenarios; {random} makes an address that cannot exist | | purpose | one sentence about what the step is for; the evaluator reads it | | lane | the row on the canvas | | readonly | this step only reads | | execution | strategy (playwright, script, agent, manual), script (path to the step script), instructions (required), requires (e.g. browser) | | mail | e-mail steps: subject and to | | criteria | what the step must meet (below) | | session | which person from sessions does this step | | affects | file globs that shape the step, for kaloko walk --affected, or always | | packs | ready-made sets of criteria: seo, usability, a11y, perf, console, errors, app, design | | capture | capture quality for this step only | | record | true records this step even when the walk does not; false never | | ignore | regions left out of pixel comparisons: { selector } or { rect: [x, y, w, h], viewport }, optionally limited by viewports, locales, with a note | | items | stack steps: where the pages come from (urls, file, sitemap with match/exclude, crawl, script) | | fields | stack steps: up to 12 columns of the stack index (id, label, from) |

Criteria

Each criterion has an id (AC1, AC12b), a text for people, an optional highlight (a CSS selector to outline on the screenshot) and exactly one check: deterministic, semantic or manual. Every check type and option is listed in Criteria, checks and packs.

Edges

| Key | What it holds | |---|---| | from, to | step ids | | label | text on the arrow, e.g. the choice at a decision | | kind | main (default) or alt for an alternative path |

Capture

| Key | What it holds | |---|---| | format | webp (default) or png (lossless, for print) | | scale | 1 or 2 (device pixels of the screenshot) | | quality | WebP quality from 0.3 to 1 | | mask | CSS selectors covered by a solid box on the image | | annotate | numbered marks on the elements of criteria with highlight | | full_page | the whole page (default) or, with false, the first screen | | record | true, step, flow, failed or off; see kaloko walk --record | | trace | keep a trace of actions, requests and console per step (default true) |

A step's capture wins over the scenario's, and the scenario's over the config's.

Design a flow with your agent

In this tutorial you design the screens of one flow as live HTML with your own agent, let the team comment on them, publish a version and make it the design that every implementation is checked against. You need a project where kaloko init has run and a scenario for the flow, for example qa/flows/checkout.yml.

Add a design environment

A design environment is a folder the CLI serves while it walks. Add it to kaloko.config.yml:

```
environments:  
design: { kind: design, serve: design }
```

The scenario stays the one development and QA use: same step ids, same criteria. A design is checked against the acceptance criteria too, before anyone builds it.

Let the agent draw the steps

Ask your agent, with whatever design skill your team uses:

> Design the checkout flow into design/ with our brand skill: one folder per step of qa/flows/checkout.yml, > shared CSS in design/_shared, tokens in design/tokens.json. Then run kaloko walk and kaloko sync.

Each step becomes design/<step>/index.html: plain HTML, CSS and JS that open from disk. Links between steps are relative links such as ../payment/index.html.

Walk, lint and sync

```
npx kaloko start --scenario qa/flows/checkout.yml --env design  
npx kaloko design lint  
npx kaloko walk  
npx kaloko sync
```

design lint checks the folder against the portable HTML rules: relative URLs, no CDNs, no storage, predictable content. walk renders only the steps that changed, and sync saves them into the shared draft and takes the steps others changed.

Picture: open the docs on Kaloko to see it

The draft of a design flow · en · desktop

Comment together

The team opens the draft on the canvas, clicks through the live steps and pins comments to elements. Everyone sees who else is looking. Your agent reads the comments with kaloko feedback, changes what they ask for and syncs again.

Picture: open the docs on Kaloko to see it

A live design step · en · desktop

Publish a version

```
npx kaloko share --name "Checkout v2"
```

This publishes a version of the whole flow, such as "v2 · 3 changed, 17 unchanged". Unchanged steps are stored once. `share --notes-preview` shows the notes on what changed before you publish.

Picture: open the docs on Kaloko to see it

A published version · en · desktop

Make it the reference

Once a person has accepted the version, make it the design reference (Team plan and up):

```
npx kaloko reference --design --version 2
```

Turn on packs: [design] for the implementation steps of the scenario. On Business and up, every implementation run is then checked against the reference: pixels, structure, tokens and contrast, and kaloko drift names the token to use where the code differs.

Coming from Figma

If the design already lives in Figma, `kaloko design import figma <file-url> --scenario qa/flows/checkout.yml` turns its frames into design steps and its variables into tokens. A folder of frames exported from Figma works without a token.

Your first run in ten minutes

By the end of this tutorial one flow of your app is on the canvas at kaloko.app, shared, with every screen in every language and screen size your scenario names. You need Node 20 or newer, Chrome, and an app running locally.

Install the CLI

In the root of your project:

```
npm install --save-dev kaloko
npx kaloko init --agent claude --org <your-org>
```

Use `--agent codex` for Codex, `--agent all` for both, or leave it out to drive the CLI yourself. `init` writes `kaloko.config.yml` and copies the skill to your agent. It also reads the project: the framework, the dev server's command and URL, the locales and the main routes. From those it drafts a first scenario, `qa/flows/main-pages.yml`, so the first walk works without editing.

If you have no organization yet, create one first; you become its admin.

Sign this machine in

```
npx kaloko login
npx kaloko doctor
```

`login` opens the browser, you approve the code and choose what the token may do. It is saved to `~/.config/kaloko/.env`. `doctor` checks Node, Chrome, the config, the environments and the token, and says what to fix next to every x.

Walk the flow

Start your dev server, then:

```
npx kaloko start --scenario qa/flows/main-pages.yml --env local
npx kaloko walk
npx kaloko preview
```

`start` creates a run, `walk` opens every step in the browser and captures it, and runs the checks as it goes. It ends with the verdicts: what failed and what a person should look at. `preview` prints the path of the local canvas, which opens from disk.

With the skill installed you can skip the typing and ask the agent instead: "Walk the main pages with Kaloko on local and show me what fails." It runs the same commands.

Share it

```
npx kaloko share
```

The run goes to kaloko.app and the command prints its link. On a branch with a pull request, `share --pr` also puts the link in a comment on the pull request.

Look at it and decide

Open the link. The run is a map of its steps; click one to see the screenshot, the criteria and what was found.

Picture: open the docs on Kaloko to see it



The map of a run · en · desktop

Approve the steps that are right, return the rest with a note, and accept the run when you are happy with it. Your agent reads the notes with kaloko feedback.

Where next

- Write a scenario for the flow you care about most.
- Review a run shows the canvas in more detail.
- Run it in CI on every pull request.
- The CLI reference lists every command.

Changelog: Kaloko

0.13.0

Released 2026-10-06

Added

- `kaloko <command> --help` (or `-h`) and `kaloko help <command>` show the help of that one command, with the options every command takes and the exit codes: 0 done, 1 checks found problems or the command failed, 2 a usage error. `kaloko --version`, `-v` and `kaloko version` print the version.
- `kaloko start --json` prints the run id and its folder, `kaloko status --json` what each step has captured.
- `--org <slug>` on every command works with another organization than `org:` in the config.
- API errors name their reason in code next to error. When the plan does not include something (HTTP 402) the answer links the organization's Plan & billing in `billing_url`, and the CLI prints that link.
- Webhooks carry `Kaloko-Signature` and `Kaloko-Event` next to `QAwalk-Signature` and `QAwalk-Event`, with the same values, so receivers written before the rename keep working.
- The trust page names a contact for reporting abuse (phishing, malware) on `kaloko.app`, `files.kaloko.app` and design previews at `*.kaloko-usercontent.com`.
- The trust page explains how Kaloko hides personal data in captures (masks, [hidden] in the stored HTML and the outline evaluators read, pii: redact) and links the how-to. The subprocessor list now names Workers AI at Cloudflare, which Clef triage uses only when an admin turns the experiment on.
- Every guide is also plain Markdown at its own address with `.md` added (`/guides/ci.md`, `/cs/pruvodci/ci.md`), for agents and answer engines. `llms-full.txt` now carries the pricing table and billing questions, the comparisons, the situations and the trust facts.
- The comparison pages answer the questions people ask about them.

Changed

- An option a command does not take gets a "Did you mean `--env?`" instead of being skipped quietly. `prune`, `trash`, `restore`, `keep`, `archive`, `public`, `release` and `comment` stop before they change anything; other commands say it and go on.
- A usage mistake (unknown command or option, missing argument) exits with code 2 and points to `kaloko help <command>`, without the support line.
- `runs`, `prune` and `screens` take `--env` like the other commands; `--environment` still works. `archive` takes `--off` like `keep` and `public` (`--undo` still works), and `prune` takes `--dry-run`, which is what it does without `--yes`.
- A service account token on a plan without service accounts is answered with HTTP 402 (it was 403): the plan, not the token, has to change.
- Webhook requests say `User-Agent: Kaloko-Webhooks/1`.
- Pricing: the print-quality row is gone, a row shows that hiding personal data works on every plan, and another that the GitHub check on the pull request comes with Team. The site says plainly that your data stays readable on every plan and that exporting a run as a ZIP comes with Team.
- Terms of service, version 4: a trial started with an invitation code needs no card; if none is added, the organization returns to Free when the trial ends. The terms and the pricing page call the billing settings Plan & billing, as the app does.
- Kaloko's public changelog and help have one address per language, which search engines and link previews now understand. The Czech changelog page says that the release notes are written in English.
- `/design` and `/cs/navrh` lead straight to the guide on designing with your agent.

Fixed

- `kaloko prune --env preview --yes` pruned the runs of every environment: `--env` was not read. It now prunes only

that environment.

- --key=value keeps everything after the first =: --map="Sign in=login" arrives whole.
- Commands run without org: in the config say how to set it instead of failing with a 404 on /orgs/undefined.
- kaloko runs says when the list stops at the newest 300 and how to narrow it.
- kaloko design import --json prints only the JSON on standard output; progress goes to standard error.

0.12.2

Released 2026-10-06

Fixed

- Czech changelog headings (Přidáno, Změněno, Opraveno, Odstraněno, Zastaralé, Bezpečnost) count as Added, Changed, Fixed and the other Keep a Changelog groups, so a Czech CHANGELOG.md is grouped like an English one.
- A form that needed a fresh second sign-in step (inviting someone, changing a setting) is sent again after the step instead of coming back empty. The fields wait in the browser tab, never on the service.

Changed

- Screens on the canvas cards stay sharp: thumbnails are 640 wide and scaled down step by step (one big jump made small text fall apart), and when you zoom in past what a thumbnail holds, the visible cards load the full screenshot. Runs shared before this get the zoom part too.
- Videos play on: once you play a step's video, the next step's video starts by itself in the step detail, and replay with video moves to the next step when a video ends (a decision waits for you, a still step shows its screenshot for three seconds). The next step's video loads while the current one plays.
- A step's video is at least 2.5 seconds long, held on its last frame, so a quick step no longer flashes past. A step where nothing moved keeps no video at all, and the canvas shows its screenshot.
- kaloko.com and www.kaloko.com lead to the same address on kaloko.app.

0.12.1

Released 2026-10-05

Added

- A trial code from us gives an organization's first subscription a longer free trial without a card. Open the sign-up link with the code, or enter it under Plan & billing. Two weeks before the trial ends, admins are asked for a card; without one the organization goes back to Free on its own and keeps its runs.

Changed

- kaloko share refuses unmasked personal data only in a run that hides it (pii on the run, scenario or environment, or mask: auto); other runs are shared with the same list as a warning. The site's own contacts no longer count: e-mails on the environment's domain, e-mails and phone numbers in the page footer, and those of an Organization or ContactPoint in JSON-LD (pii: { site_contacts: false } checks them again). Sample addresses such as you@company.com, placeholder text and SVG drawings are not read as personal data.
- Phone numbers written 603 123 456 count only next to a word like Telefon or Mobil, and amounts, EANs and order numbers no longer read as phone numbers. The nine-digit birth number needs its label (RČ), as does one without the slash.
- pii: pseudonymize gives one value the same stand-in in every run of the scenario, so screenshots and approvals carry over, and two values never share one. KALOKO_PII_SALT adds a secret of your own.

Fixed

- Hidden form fields, value attributes and personal data in URLs (also encoded, jan%40firma.cz) are masked.
- The capture record no longer keeps real data: page title, headings, JSON-LD, microdata, console lines, blocked requests, the URL and check evidence all go through the mask, and a capture whose masked page

cannot be read stores nothing.

- Masking no longer changes what the page reacts to: console errors are read before it, data-* attributes are masked only in the stored HTML, and the keyboard walk runs on the real page with masked focus shots.
- Short masked values such as 2024 or Praha are no longer replaced across the whole run (width:100% stays).
- Recordings cover personal data in modal dialogs and popovers, open shadow DOM and same-origin frames, and on a new page from its first frame with everything masked so far. Screenshots of steps mask the same places.
- Values masked in one kaloko process are masked in the next process of the run too.
- Triage (experimental) fits long pages into Clef: two page tiles and smaller recording frames instead of four full tiles, and a request still too large is asked again with half the images. A Cloudflare token in the project that cannot use Workers AI hands triage to Kaloko instead of stopping it, and such a token is used only when `evaluators.triage` is set in `kaloko.config.yml`.
- A failure while masking leaves the page as it was.
- Long words without spaces no longer slow down detection.

0.12.0

Released 2026-10-05

Added

- Triage, an experiment an admin switches on in Settings -> Security -> Experiments. kaloko evaluate shows each screen, and a few frames of its recording, to Clef on Cloudflare, which names the kind of problem it sees: overlapping elements, cut-off text, a raw translation key, a placeholder, an error, an endless spinner, a layout that jumps. The canvas lists them under "Possible problems"; one click turns a right one into an ordinary comment, and findings never change a verdict. `evaluators.triage` in `kaloko.config.yml` tunes or turns it off.
- MCP tools `get_findings` and `answer_finding`.
- Masks hide personal data in everything a step stores, not only in the screenshot. The captured HTML and the page outline the evaluators read `get [hidden]` of the same length (`[skryto]` in Czech runs), and traces, recordings, crops, focus shots and the HTML view are covered too. Checks still read the real page, so their verdicts do not change.
- pii: redact on a scenario finds e-mails, phone numbers, birth numbers, IBANs, account numbers, dates of birth, addresses and names in greetings without a selector, in Czech formats too; capture: `{ mask: auto }` does it for one step. `pii.allow` in `kaloko.config.yml` lists values that only look personal, such as your support line.
- pii: pseudonymize replaces personal data with made-up values instead of boxes. One value gets the same stand-in for the whole run, so you can still see a name travel from the form to the summary and the e-mail.
- Captured e-mails hide their recipients (To and Cc, now captured too) whenever a step masks anything; greetings and payer blocks follow the step's masks.
- kaloko share checks the run before it uploads: an unmasked e-mail, phone number, birth number or IBAN stops it, with the step and the element. `--allow-pii` uploads anyway.
- An environment with `pii: required` gets no run that would keep personal data; kaloko start `--pii redact` (or `pii: on` on the scenario or the environment) satisfies it.
- The canvas says "Personal data hidden" on such runs and outlines the masked places in the step detail, so a dark box is not taken for a bug in the app.

Changed

- The masks of a scenario and of its step now add up; before, a step's `capture.mask` replaced the scenario's.
- `kaloko.app` serves its own fonts and the canvas you open from disk carries them inside, so no page asks Google Fonts for anything and Google Fonts is off the subprocessor list on the trust page.

0.11.0

Released 2026-10-04

Added

- The approval page of kaloko login says where the request came from (city, country and address), which terminal asked and when, and starts with a plain warning: approve only a sign-in you started yourself, just now. When the request came from another country than the one you are in, the page says so in red.
- SCIM tokens are valid for a year. Settings -> Security -> SCIM shows the date, and "Renew for a year" keeps the same token, so nothing changes in your identity provider. Admins get a reminder in the inbox 30 days before a token expires. Tokens you already have run for a year from this release.
- A company sign-in over SAML set up before "Require a signed response" existed turns it on with one click in Settings -> Security. Kaloko offers the button once it has seen your identity provider sign its responses, so people keep signing in as before.

Changed

- A passkey used as the second step must ask for your PIN, fingerprint or face. A security key that signs without asking no longer counts; your account page names the key and says how to bring it back (set a PIN on it, or add another passkey).
- A run whose title, environment, branch, commit message or another field is too long is refused at the start of the upload with a message that names the field and the limit, instead of an unexpected error halfway through.

Fixed

- A browser that launched but then stopped answering no longer holds kaloko walk, evaluate --recheck or a capture until someone kills it. A new browser must open its first window within 20 seconds or it is closed and started once more, and later windows have the same limit. kaloko browser start, kaloko doctor, kaloko auth save, the PDF export and connecting to the shared browser now have the time limit and the second try too (KALOKO_LAUNCH_TIMEOUT, seconds).
- A js_equals check on a busy machine no longer fails when the page answers a little late: the expression keeps its 2-second limit, and Kaloko waits up to 17 seconds for the page to reply.

Security

- The sign-in cookie can be set only by kaloko.app itself. You stay signed in while it changes over. Your session also gets a new id at every sign-in and right after the second step.
- App and website pages tell the browser which features they never use (camera, microphone, location, payments and the like), and a Kaloko tab opened from another site is cut off from that site's window.
- Billing events from Stripe and pull request events from GitHub are applied once, also when someone sends a captured copy again.

0.10.0

Released 2026-10-04

Added

- A step can list its other states: states: [empty, error, loading]. Walks run the step script once per state (it gets state), a design draws cart/empty.html next to cart/index.html, and each state gets its own screenshots and verdicts. The canvas has a State switcher next to the language, and review mode and the grid show the state you pick. A criterion with states: [empty] is judged on the empty cart only.
- A design whose token set has several modes, such as two brands, is rendered in each of them. Switch modes on the canvas like colour schemes; tokens_mode in the environment stays the default. Light and dark alone still follow color_schemes. Variants of a stack step are captured in every state and mode too.
- kaloko capture --state, kaloko spec --state and a state option on the MCP step tools (get_step, get_step_spec, get_step_drift, get_step_assets, get_step_recording).

- A design step can come in variants of one template, such as an e-mail for each reason support can pick. Make it a stack step and give each variant a folder, `design/<step>/<variant>/index.html`. kaloko walk captures every variant in every viewport, kaloko sync sends them to the draft, and in a published version the step opens a list of its variants with their results. Each variant opens live. kaloko design lint reports a variant folder without its page, and both design exports include the variant folders.
- Web fonts on live design steps. An organization admin lists the addresses designs may take fonts, stylesheets and images from (Google Fonts, Adobe Fonts, your CDN) in Settings -> Security, and live steps load them instead of falling back to a system font. Scripts still load only from the design itself. List the addresses under origins of the design environment and kaloko design lint says which of them the organization does not allow yet.
- Prototypes remember things between steps: a cart filled on one step shows on the next, in Live mode and in the presentation. The state stays in the viewer's browser tab, every step opened alone still shows its own sample data, and "Start over" in the presentation clears it. Design agents use `window.kaloko.state`.

Changed

- Large runs are quicker on the canvas. On a run of 500 steps in seven languages on three devices, with the CPU slowed down to a phone's, a zoom step takes 33 ms instead of 133 ms, switching between desktop and mobile 650 ms instead of 910 ms, and the map is drawn once when it opens (it was drawn twice). Scrolling the step index no longer sorts every step on each frame.
- On `kaloko.app` a large run's canvas data and a scenario's history page come from a cache after the first visit (27 ms instead of 113 ms, 9 ms instead of 230 ms), and the changelog, the products page, the screen library and review mode ask the database far fewer times.

0.9.0

Released 2026-10-04

Added

- After a release, people who read the product see a short "What's new" bar in the app that links its illustrated changelog. It comes once per release and person; open the changelog or close the bar and it stays gone.
- The canvas tells you when the connection drops or Kaloko cannot save. Comments, approvals and verdicts wait and go out once you are back online, or with "Try again"; nothing is posted twice and nothing is lost quietly.
- `kaloko doctor --fix` installs a missing browser, creates missing folders and fills in org: from your token. An expired token points you to kaloko login.
- Long stack walks show roughly how much time is left.

Changed

- On a phone, review mode puts Approve and Return under your thumb at the bottom of the screen and says what you just decided and which step comes next.
- The first-steps list links straight to where each step is done: your newest run for comments and approvals, with the command to copy for the steps done in the terminal.
- Every error page has a way to write to support, with the address already in the e-mail, and a link to the status page. The app and website footers link the status page too.
- CLI errors say what happened and what to do next (sign in again, wait and retry, check the network) and end with `support@sinfin.cz`. Losing the connection no longer prints a stack trace.

Fixed

- A browser video encoder that stops responding no longer holds kaloko walk `--record`: the recording ends with a note and the walk goes on.
- Dark mode is easier to read: form fields have visible edges, the highlighted plan column on the pricing page is readable, and shortcut keys on the canvas's review buttons stand out.

0.8.0

Released 2026-10-04

Added

- Steps and scenarios say what kind of page they are: page: landing, work, list, settings, form or doc. The usability pack asks questions that fit: "one clear primary action" only on landing pages, and on lists, settings, forms and docs a question of their own. Without page, public pages count as landing pages and app screens as work pages, so a signed-in overview is no longer asked for a single call to action.
- A criterion can apply to some environments only: environments: [production] on hreflang pairs, a live demo or the production sitemap. On a local or staging walk it shows as not applicable to that environment and does not make the step partial.
- kaloko validate warns when a scenario walks several languages and a text check matches only one of them, for example pattern: "Plans and payment" without the Czech wording.
- Empty pages in the app say what to do next and give the command with a copy button: a product without a release or docs, its changelog and docs, an organization without runs, the runs and projects lists, the inbox and the screen library. People who only read a product see a plain sentence instead of a command.
- Kaloko's help covers the whole product, in English and Czech: two tutorials, how-to guides from writing a scenario to company sign-in, a reference of every CLI command, scenario key, check, pack, config key and MCP tool, and explanations of how verdicts are decided and how Kaloko maps to your applications.
- The website links Kaloko's own public changelog and help, so you can see both working before you set them up.

Picture: open the docs on Kaloko to see it

The home page · en · desktop

- Company sign-in over SAML can require a signed response: the identity provider must sign the whole response, not only the assertion. New connections start with it on. Connections set up earlier keep working as before, and Settings -> Security recommends turning it on.

Changed

- Error pages say why you got there and how to go on. A missing page offers the overview or another sign-in; a page that needs a role you do not have offers an e-mail to your organization's admins, already written; an expired link says to ask for a new one; an ended session signs you in and brings you back to the same address.
- Unsent text on the canvas survives. The verdict, return and issue notes keep their draft like comments do, and when your session expires while you write, the canvas says so, keeps the text and returns you to the same step after you sign in again.

Fixed

- A browser that starts but never answers no longer leaves kaloko walk, video or a check waiting: each launch has a time limit and one more try (KALOKO_LAUNCH_TIMEOUT in seconds, default 45).

- kaloko init proposes products from npm, Yarn and pnpm workspaces and from packages/ too, not only from apps/; shared libraries without a dev, start or serve script are left out.
- kaloko stability --pull names the button people use on kaloko.app: Mark a changing region.
- A table in published docs can hold a pipe inside a cell, written as `\|`.
- Hosted runs open only pages whose address leads to the public internet. An address that points into a private network is refused when you schedule the run and again before each run. Saved passwords and cookies are sent only to https addresses.
- Text and attribute checks in hosted runs finish quickly whatever the pattern. Patterns with backreferences or lookahead are reported as not evaluated there, with the reason; kaloko walk still checks them in full.
- Signing in, sign-up, invitation links and the second step have their own limits on repeated attempts. Exports and new API tokens have a generous hourly or daily limit per person.

0.7.1

Released 2026-10-04

Fixed

- kaloko walk of an Android, iOS, desktop or Electron app ends with the verdicts, like a walk of a website; before, the criteria stayed unchecked until you ran kaloko evaluate.

0.7.0

Released 2026-10-04

Added

- Products in the app. The main menu has Products: every product you read with its newest version, its docs, its modules and projects, who reads it and links to its changelog and docs. A project page names the products it belongs to. kaloko products sync puts the products of kaloko.config.yml on Kaloko before their first release, and kaloko products list shows which ones are there.

Picture: open the docs on Kaloko to see it

The Products page · en · desktop

- A trust page at kaloko.app/trust (in Czech at /cs/duvera): where your data lives, how it is encrypted, who can get in, which companies process it and where our SOC 2 preparation stands. Kaloko is not SOC 2 certified yet, and the page says so. The security contact is also in /.well-known/security.txt.

Picture: open the docs on Kaloko to see it

The trust page · en · desktop

- Seat warnings. When 80 % of the paid seats are taken, and again when all are, organization admins see a banner on the home page, in Settings and on the billing page, and get one e-mail per threshold in each billing period. GET /api/v1/orgs/<org>/usage reports the seats too.
- Yearly Enterprise subscriptions can go above their paid seats. Kaloko keeps the highest number of seats in each quarter; after the quarter, Kaloko billing reviews the true-up and invoices the extra seats for the rest of the subscription year or waives them. Seats added in the last quarter are not charged for that year: a week before the renewal the subscription rises to them and the renewal invoice includes them. The billing page shows the current quarter and every earlier true-up.
- Company sign-in over SAML accepts encrypted assertions. Create the organization's key pair under Settings -> Security -> Company sign-in; its certificate is in the SP metadata, and a new key pair keeps the old one working until you remove it.
- Single logout with your identity provider: signing out at the provider signs people out of Kaloko, and signing out of Kaloko can sign them out at the provider too.
- Sign-in from the app tile of your identity provider, off until an admin turns it on, with a default page people land on.

Fixed

- The keyboard checks of kaloko walk no longer report a trap inside a third-party widget such as an anti-bot check, and judge a link wrapped over two lines and a small radio inside its label as people see them. Tab lists with arrow keys count as one tab stop.
- The canvas works better from the keyboard: every control shows a clear focus ring in light and dark mode, a "Skip to content" link jumps past the top bar, and while a step detail or another overlay is open, Tab stays in it instead of wandering through the hidden canvas. Screen readers get named side panels and dialogs.
- The canvas no longer jumps while it loads.
- A link from a comment notification opens the step with its comments in view, on a phone too.
- On a phone, a live page in the step detail fills the room above the sheet instead of a short strip, an open sheet leaves a strip of the preview that you can tap to lower it, and "Ctrl+Enter sends" shows only with a keyboard.

0.6.0

Released 2026-10-03

Added

- A visual changelog. kaloko changelog draft writes the Unreleased entry of CHANGELOG.md from the pull requests since the last release and adds before and after pictures of the screens your accepted runs show differently. kaloko release <version> records the release with its pictures, and kaloko.app shows it as a page you can share with clients, support or anyone outside development.

- Step references in Markdown: always shows the newest accepted screen, and ?version=2.4.0 the screen of that release.

No accepted picture yet

kaloko:checkout/payment

- Products and modules in kaloko.config.yml for repositories with several apps; scenarios say which product they show (product:, module:). Without them the project is the product.
- Readers from your domain: everyone who signs in from a verified domain reads the changelogs, also of restricted projects (Business and up).
- kaloko init asks whether to keep a visual changelog, kaloko doctor checks that every product's version can be read, and agents get get_changelog, draft_changelog_entry and resolve_step_image over MCP.
- Docs that keep up with the app. Guides in your docs/ folder show steps instead of screenshots; kaloko docs publish puts them on kaloko.app with a version picker, language and screen switches and search, and every picture opens its step on the canvas.

Picture: open the docs on Kaloko to see it

A guide with its pictures · en · desktop

- When a screen changes after its guide was written, the section says "the screen changed in 2.4 — check the text", and kaloko docs check lists it with broken references, missing translations and changelog items without a picture, so CI can stop on them.
- A guide plays as a walkthrough, one step at a time with its words as the caption, and readers tick off the steps they tried.
- Exports for your own docs site: kaloko export --format md, docusaurus, vitepress, mkdocs, html, pdf or json, with the pictures as files. The docs and changelog pages offer the same downloads.
- A "What's new" e-mail: readers ask for it on a product's changelog and get each new release with its pictures in their language (Business and up).
- kaloko ci github --release records every release that release-please, changesets or semantic-release

publishes.

- Issues in Jira and Linear from a returned step or a failing criterion, on the canvas, in review mode or with kaloko issue create. The issue gets the screenshot, the criterion, the notes and a link back, and the comment on the step is resolved when the issue is done. Branches and pull request titles that name an issue link the run to it.
- Slack and Teams channels per project, each with the events it wants and quiet hours. Restricted projects reach only channels chosen for them.
- A short list of first steps on the home page (share, comment, approve, invite, CI, release) until the team has done them.
- Hosted runs: Kaloko opens the pages of a read-only scenario every day or week and shares the run (kaloko schedule add, Business). kaloko schedule export github writes a scheduled workflow for the full walk.

0.5.2

Released 2026-10-03

Added

- The first version of the visual changelog: kaloko changelog draft, kaloko release, step references and the changelog page on kaloko.app with a before and after slider. It is described in full under 0.6.0.

0.5.1

Released 2026-10-03

Added

- Swipe and onion skin compare next to side by side and diff, in the step detail and in review mode.
- Regions to ignore can be drawn by hand on a capture. The canvas diff skips them, and kaloko stability --pull writes them into the scenario.
- A grid (key G) shows one step in every language and screen size at once, also on a phone and in review mode.
- Comments can carry drawn boxes, arrows and lines, and attached images. Agents get the marked region in pixels.
- An accepted run has a signed acceptance record: a PDF with thumbnails and a printable page. kaloko signoff --verify checks the signature.

0.5.0

Released 2026-10-03

Added

- Checks of the page itself: the address a step ends on, computed styles of an element and values the page exposes, so scenarios no longer need data-qa markers.
- Runs in WebKit and Firefox besides Chromium, on device presets (phones, foldables, tablets, TV), in dark mode, with reduced motion, forced colours or as an installed web app. Every step is captured in each combination.
- App walks on several devices at once: Android emulators, iOS simulators, a real iPhone and Android WebViews, and Windows apps next to macOS ones.
- One picker on the canvas for language, browser and colour scheme that stays usable with many values.

Picture: open the docs on Kaloko to see it



The canvas of a run · en · desktop

- Workflows written by kaloko ci install the browser engines your scenarios walk.

Fixed

- Design drafts keep the browsers, schemes and device presets of their scope.
- An e-mail captured during a walk in several browsers counts once.
- Language and theme switchers work in the phone menu.
- Ignored regions, imported screenshots and cached evaluator answers carry over between capture variants.

0.4.11

Released 2026-10-03

Added

- Recordings: kaloko walk --record films each step from the previous screen to its capture, or the whole flow with a chapter per step, and keeps a trace of actions, requests and console messages for every step.
- The canvas plays a step's recording and shows its trace.

Picture: open the docs on Kaloko to see it



A recording on the canvas · en · desktop

- kaloko video export turns the recordings into one walkthrough video with a title card per step.
- Shared runs carry their recordings on the Business and Enterprise plans, kept for 30 days unless the organization chooses otherwise.

Fixed

- A recording too large to upload stays in the local preview instead of failing the share.

0.4.10

Released 2026-10-03

Added

- `kaloko ci github|gitlab` writes a workflow that walks what each pull request changes on its preview deployment and comments on the pull request; `kaloko ci preview-url` finds the preview address in CI.
- `kaloko import` lays out screenshots, recordings and traces that an agent or Playwright already produced as a run.
- `kaloko walk --affected` and `kaloko affected walk` only the steps that the changed files reach, and say why.
- `kaloko walk --heal` proposes a new selector when the page lost the old one, and applies it once the step passes.
- Several languages walk at the same time (`--concurrency`).
- `kaloko init` recognises the framework, the dev server, languages and main routes, and drafts a first scenario.
- Uploads skip screenshots the service already has and resume after a broken connection.

0.4.9

Released 2026-10-03

Added

- A vision evaluator that looks at the screenshot and says why, keyboard and screen reader checks, and regions that change on every run can be ignored.
- Dev mode in the step detail of a design: box model, tokens, redlines, states and assets.

Picture: open the docs on Kaloko to see it

Dev mode in the step detail · en · desktop

- `kaloko spec` tells an agent what to build for an approved design step, and `kaloko drift` what its implementation does differently.

Fixed

- The canvas on phones: a smaller stack badge, readable stack lists, the step header above its tabs and no sideways scrolling.

Picture: open the docs on Kaloko to see it



A stack page on a phone · en · mobile

- Page counts in Czech use the right plural.

0.4.8

Released 2026-10-03

Added

- Review mode: the canvas walks a person through what still needs a decision, with keyboard shortcuts, and kaloko review lists the same for agents.
- Approvals carry over to steps whose screenshots did not change since the last accepted run.
- Every new organization gets a sample run with a guided first review.
- Two-step verification with passkeys or an authenticator app, and an organization rule that requires it.
- An audit log for admins, tokens limited to chosen projects, and company sign-in with SAML or OIDC, SCIM and an IP allowlist on Enterprise.

Fixed

- Plan badges on the pricing page no longer overflow on phones.

0.4.7

Released 2026-10-03

Added

- Sign in with Google, Microsoft or GitHub where the service offers it.

Picture: open the docs on Kaloko to see it



The sign-in page · en · desktop

- kaloko login approves a token in the browser instead of copying it from Settings.
- Invitations: people outside the organization join single projects as guests, and the Share dialog of a project shows who has access.

0.4.5

Released 2026-10-03

Added

- A Designer role and API tokens with chosen scopes (read, comment, design, upload).
- Focus mode shows the preview alone, and the step detail has a layout for phones.

0.4.4

Released 2026-10-02

Fixed

- kaloko evaluate --recheck says that it asks the semantic questions again too.
- A lane on the canvas is as tall as its tallest card.

0.4.3

Released 2026-10-02

Fixed

- Signed-in pages that Kaloko's own walk found wanting: labelled fields, real settings tabs, cards on phones and the whole inbox.

Picture: open the docs on Kaloko to see it



Settings · en · desktop

- The canvas has a main landmark, readable badges and a visible compare button, and no longer shifts while loading.
- Pages load their fonts sooner and keep their layout when the font arrives.

0.4.2

Released 2026-10-02

Added

- The Kaloko wordmark with a yellow marker and a new icon.

Picture: open the docs on Kaloko to see it

The home page · en · desktop

- The Czech website lives under /cs/, so each language has its own address.
- Pages of a stack may carry their own purpose for the semantic evaluator.
- Live design revisions open on their own domain.

Fixed

- kaloko doctor prints the config file it actually read.

0.4.1

Released 2026-10-02

Added

- links_ok and hreflang_pairs checks for pages and stacks.

Fixed

- Unknown addresses on kaloko.app answer 404 instead of the sign-in form.
- Every language version of the website answers at its own address.

0.4.0

Released 2026-10-02

Changed

- QAwalk is now Kaloko: the CLI is kaloko (the qawalk command and qawalk.config.yml keep working) and the service lives at kaloko.app.

Picture: open the docs on Kaloko to see it

The landing page · en · desktop

Added

- Kaloko Design presents interactive HTML designs, lints pages that must work from disk, and exports a design in open formats.

0.3.0

Released 2026-10-02

Changed

- The CLI is on npm as kaloko instead of @qawalk/cli. The qawalk command, qawalk.config.yml, the QAWALK_* variables and ~/.config/qawalk/.env keep working, and kaloko init replaces the old skill.
- The app and the website share kaloko.app. Pages on qawalk.com redirect there; the API, MCP, webhooks and file links on the old addresses keep answering.

Added

- A "Group by issue" switch on the lists of runs and scenarios. The lists remember it, and "Latest per scenario" too.

0.2.22

Released 2026-09-30

Fixed

- Large runs open faster on the canvas, and kaloko evaluate needs less memory for big stacks.

0.2.21

Released 2026-09-30

Fixed

- A kept run cannot be trashed, and releasing it gives it at least a week. Kept design versions are not pruned.
- The "no project" filter on runs works, failed step details load again, and an unusual cookie no longer breaks the lists.
- Lists, the home page and scenario history load faster.

0.2.20

Released 2026-09-30

Added

- The project you pick is remembered across the lists and the scenario browser on the canvas.
- Scenarios and runs of one issue, or of one pull request when there is no issue, are listed together.

0.2.19

Released 2026-09-30

Changed

- Shortcuts and help open from a keyboard button in the top bar, and the list now includes <- -> and Esc.

0.2.18

Released 2026-09-29

Changed

- A shorter top bar on the canvas: Projects, Scenarios and Runs, the view switch, and the inbox at the right end.

0.2.17

Released 2026-09-29

Changed

- Projects, scenarios and runs share one layout with the navigation on the left, so switching views does not jump. "All runs" is now "Runs".
- The run verdict has Accept and Return as the main buttons; note, baseline and public sit below them.

0.2.16

Released 2026-09-29

Added

- A projects page: scenarios, runs and how the newest runs stand in each project.
- Console errors in the step detail are highlighted, wrapped and can be copied one by one or all at once.
- Times follow the reader's time zone, set in Settings -> Profile or taken from the browser.

Changed

- Screen size switches say the name and width (Desktop 1440, Mobile 360), and a click anywhere on a card opens the step.

0.2.15

Released 2026-09-29

Changed

- The step detail has three tabs: Criteria, Page and Technical. Criteria that need attention come first; passing ones take one line.
- When only phones are shown, the map shows larger phone screenshots.

0.2.14

Released 2026-09-29

Changed

- Evidence of built-in checks reads as a sentence in Czech or English ("TTFB 950 ms, the limit is 800 ms"); the raw text stays under Technical detail.

0.2.13

Released 2026-09-29

Added

- Every criterion of the built-in packs says why it matters and how to fix a failure, in Czech and English. The walk summary prints the fix for the agent.

0.2.12

Released 2026-09-29

Changed

- The run summary sorts problems by what to do: what failed and why, what a person should decide, and screens that look off.

0.2.11

Released 2026-09-29

Added

- Quick signals on every screen: the evaluator also says whether the page does what it is for, is in the expected language or looks broken. Signals never change a verdict.
- kaloko walk evaluates while it walks and ends with a list of what failed or is unsure.

0.2.10

Released 2026-09-29

Added

- Every criterion says how it was checked, in words: the check as a sentence, or the question the AI was asked with its score and the pass threshold.
- Hosted AI evaluation from the Team plan up, with a monthly allowance shown on the billing page and in kaloko doctor. Your own evaluator key still works.
- Design tokens in the current DTCG format, with aliases, modes and deprecated tokens; kaloko tokens validate.

Fixed

- Manual decisions show the criterion on its own line, with reason and verdict below.

0.2.9

Released 2026-09-29

Added

- Scenarios come first in the navigation, and Runs show the newest run per scenario unless you filter.
- Keep a run so it never expires, prune old runs (dry run first) and archive scenarios you no longer walk, in the app or with kaloko runs, trash, restore, prune, keep and archive.

0.2.8

Released 2026-09-29

Changed

- A claimed sign-in domain lets people join only once it is verified.
- Comments on public runs name nobody, and upload tokens cannot delete runs.
- Below 1500 px the navigation and view options fold into one menu.

Fixed

- The canvas shows the map sooner and loads step details after it; large runs upload faster.
- When the service asks the CLI to slow down, it waits and tries again.

0.2.7

Released 2026-09-29

Added

- Comments on steps, whole runs and whole flows: threads, pins on the screenshot, resolve and reopen. Mentions are mailed at once, the rest goes to the digest, and agents use kaloko comment or MCP.
- Kaloko Design: design drafts as live HTML steps on the canvas, versions, an inspector with token names, comments pinned to elements, edits on the canvas, branches linked to git, and a design pack that checks an implementation against the approved design.

- An HTML view of captured pages (experimental) that can be forked into a design draft.

Changed

- A calmer run view on smaller screens: one top bar, view options in a popover, help behind ?.

Fixed

- A long step path no longer widens its card over the next one.
- macOS desktop captures ask for the Screen Recording and Accessibility permissions they need.

0.2.6

Released 2026-09-29

Added

- Mobile and desktop apps: Android, iOS Simulator, Electron and macOS windows, with UI tree checks, touch target sizes, crash detection and an app pack. kaloko capture --image imports screens from other tools.
- A run index next to the map (key I): every step in a list with filters, search and CSV, and stacks as folders.

0.2.5

Released 2026-09-29

Added

- The HTTP status of every capture, an errors pack for error pages, and {random} in a path for an address that cannot exist. kaloko draft adds a not-found step.
- The run card links the issue next to the pull request.

0.2.4

Released 2026-09-29

Added

- Stacks: one step stands for many pages of one template, taken from a list, a sitemap, a crawl or a script. The canvas shows them as a stack with an index you can filter, search, sort and export as CSV.

Picture: open the docs on Kaloko to see it

A stack opened as a folder · en · desktop

0.2.3

Released 2026-09-28

Added

- acceptDialogs() for step scripts, flags on text checks, kaloko evaluate --recheck after a scenario fix, and kaloko

walk --ephemeral for parallel walks.

Fixed

- kaloko mail reads only messages received since the run started.
- A walk captures fresh screens after --steps, basic auth survives a redirect, and only one walk at a time uses a shared browser.

0.2.2

Released 2026-09-28

Added

- kaloko draft <url...> writes a first scenario and acceptance plan for public pages.
- Check packs a11y, perf and console.
- Live presence on the canvas: who is looking, their cursors, and following someone's view.
- Large flows stay readable: lanes with headers, long lanes wrapped into rows, tidy edges and a zoomed-out view.
- Scenario history with pass rate and criterion stability, and kaloko calibrate to tune evaluator thresholds.
- Print-quality captures for documentation, with masks over personal data and numbered marks.
- The newest capture of every step at a stable address, a list of changes, signed webhooks and kaloko export --scenario for documentation (Business and up).
- GitHub checks on pull requests that complete when the run is accepted or returned.
- A sign-in domain can be proved by signing in with Google Workspace or Microsoft 365, and Settings shows the DNS host with copyable record fields.

Fixed

- The overview follows reviewers' criterion verdicts.
- Public mailbox domains such as gmail.com cannot be claimed.
- A walk stops a session after three steps fail with the same error, instead of trying a wrong password again and again.
- An open menu or popover stays on the capture, signed links in the page lose their credentials, and text check evidence shows the actual match.
- kaloko share says when the plan grants a shorter lifetime than requested.

0.2.1

Released 2026-09-25

Added

- Protected environments: HTTP Basic, service-token headers and client certificates, sent only to the environment's own addresses.
- Accounts per scenario with password and TOTP, and kaloko auth save for a single sign-on a person does once.
- \${VAR} in base_url for review apps, and secrets named in the config are removed from captures.
- Any mailbox through a script adapter, and kaloko mail --file for saved messages.

0.2.0

Released 2026-09-25

Added

- The CLI is on npm (as @qawalk/cli until 0.3.0).

- Plans with a billing page, a 14-day trial and a public pricing page.
- Named sessions: several people in one flow, each in their own browser, shown in swimlanes on the canvas.
- Run verdicts, a baseline run, required approvers and reviewer verdicts for manual criteria.
- Compare a run with the baseline or any earlier run: pixel differences, changed criteria and evidence, and kaloko compare.
- Replay walks the run one step at a time with a branch choice at decisions, and the canvas has a minimap.
- An inbox with notifications, a daily or weekly digest, and Slack.
- kaloko share --pr comments on the pull request; kaloko export and the canvas download a run as a ZIP.
- An MCP server for agents.
- Check packs seo and usability, plan coverage with kaloko validate --coverage, and flaky criteria.
- A public demo run anyone can open without signing in.
- Projects per scenario, and a runs list filtered by project, scenario, verdict, author, environment and branch.
- Sign-in domains verified by a DNS record, token scopes and expiry, a list of your sessions, and a 7-day trash.
- kaloko doctor checks your setup and says what to fix.
- Guides on the website for use cases and agents.

0.1.0

Released 2026-09-23

Added

- The first version. The CLI walks scenarios in Chrome, captures every step, runs the checks and asks an AI evaluator what a check cannot answer. A read-only policy keeps production safe.
- The canvas shows the run as a map of screens with criteria, page metadata (Open Graph, hreflang, structured data, headings) and links to a single step. A click on a criterion highlights it on the screenshot.
- The agent skill for Claude Code and Codex, installed with kaloko init --agent.
- The service: organizations by e-mail domain, sign-in by link, roles and API tokens. kaloko share uploads a run, reviewers approve or comment on each step, and kaloko feedback brings their notes back to the agent.
- The app and the website in Czech and English, with sign-up for new organizations.