

Move a GitHub repository to an organization without breaking Cloudflare Pages

Transferring a repository that Cloudflare Pages builds and serves. Covers what GitHub moves and what it leaves behind, recording the project settings first, preparing the organization, rebuilding the Pages project, and moving the custom domain.

GITHUB · CLOUDFLARE PAGES · REPOSITORY TRANSFER · HOW-TO GUIDE

General educational guidance, not affiliated with or endorsed by GitHub or Cloudflare. Describes both platforms as of September 2026. Published 2026-10-01.

10 steps, planning to verification

the transfer itself is one confirmation; rebuilding the Pages project is the part that needs planning

2 documented recovery options

once the Git link breaks: transfer the repository back, or build a new Pages project; the dashboard has no reconnect option

SCOPE What this guide covers, and what to know before you start

Moving a repository from your personal GitHub account into an organization hands the code over to the company. If Cloudflare Pages builds and serves your site from that repository, the move breaks the link between them. The Pages project keeps a reference to the old owner, and the Cloudflare dashboard offers no way to update it. This guide rebuilds the Pages project around the moved repository. Moving a repository between two organizations breaks the same link, so the steps apply there too, with the differences Step 1 notes. The guide describes both platforms' documentation as it stood in September 2026; where a detail comes from the two dashboards as they appeared then, it says so.

Where to start

- **Not transferred yet:** read Step 1, do Steps 2 and 3, then transfer the repository in Step 4.
- **Already transferred, and the Pages project now warns that deployments may fail:** your site is still up. Go to the last section, "If the transfer already happened".

Your live site stays up, except while its domain moves

Cloudflare serves your site from deployments already uploaded to its network, and the Git connection only triggers new builds. When the transfer breaks that connection, the dashboard warns that deployments may fail, but visitors keep getting the last deployment on every domain the project serves.

The one planned outage is moving the custom domain in Step 8. The domain is unreachable from the moment you delete its CNAME record (the DNS entry that points your domain at the Pages project) until Cloudflare has issued the domain's new certificate and marked it active. Where the domain's DNS is with another provider, the outage starts instead when you remove the domain from the old project. Plan for at least 15 minutes. Deleting the old project in Step 9 also takes down anything still pointing at the old project's `pages.dev` address (the `<project>.pages.dev` hostname Cloudflare gives every project) or at a domain Step 8 did not move.

The Pages project will not follow the repository

Cloudflare's documentation gives two options for a transferred repository: transfer it back, or create a new Pages project at the new location. The GitHub App that gives Cloudflare access is installed separately on each GitHub account, so the installation on your personal account never covers the organization. Reinstalling the app on the organization does not repair the old project either, because the project keeps its stored reference to the old owner.

Cloudflare's API schema (the machine-readable description of its API) also lists a call that disconnects a project's Git source and one that connects a source to an existing project. The Cloudflare CLI (its command-line tool), in beta since 28 September 2026, has a command for each: `cf pages source disconnect` and `cf pages source connect`. Neither call appears in Cloudflare's API reference, its SDKs (its code libraries for developers), or its Pages documentation. This guide has not tested them on a project in this state, so it does not rely on them, and neither should you on a live site.

Steps 2 and 3 come before the transfer

Step 2 records the Pages settings that live only in Cloudflare's dashboard, because the old project is the only record of them. Step 3 locks down the organization before the repository arrives, because the organization's default permissions apply to the repository the moment the transfer lands. If you have already transferred, both still work: the old project keeps its settings until you delete it.

01

Know what the transfer moves, and what it leaves behind

GitHub's transfer carries almost everything with the repository, including the people you had already given access. It leaves behind every GitHub App installation. For a private repository moving to a free organization, it also drops paid-plan protections such as protected branches.

WHAT	AFTER THE TRANSFER
Git data, releases, tags	All move. Commit history, contributions, tags, and the releases built on them arrive intact.
Issues, pull requests, wiki, stars, watchers	All move. On a move from a personal account into an organization, issue assignments to organization members survive and all others are cleared. GitHub also says only organization owners can create new issue assignments.
Webhooks, deploy keys, secrets	Stay with the repository. Cloudflare's app adds no repository webhook of its own: like any GitHub App, it receives push events through its installation, which Step 6 replaces.
Visibility and plan features	A private repository stays private. GitHub's transfer documentation says a private repository transferred to a free account or organization loses paid-plan features. It names protected branches and GitHub Pages (GitHub's own site hosting, separate from Cloudflare Pages) as examples, so the full list may be longer. Check what protections you had, and pick the organization's plan before you move (Step 3).
Packages, Git LFS files	Git Large File Storage (LFS) objects move in the background, so a large store keeps transferring after the repository itself has landed. Packages depend on the registry they are published to: some move with the repository, others lose their link to it. Check any registry you publish to.
GitHub App installations	Stay behind. GitHub's transfer page does not mention apps. The rule follows from how installations work: an app is installed on one account or organization and reaches only that account's repositories. Every app that could reach the repository under your account loses access to it. Step 6 installs the Cloudflare app on the organization.

WHAT	AFTER THE TRANSFER
Permissions	The organization's default repository permissions and member privileges apply the moment the repository arrives, on top of the collaborators who already had access, so Step 3 sets them first. GitHub also adds the previous owner, your personal account, as a collaborator on the transferred repository. As an organization owner you already have access, so you can remove that entry.

Only for a move between two organizations: you also need permission to move repositories out of the origin organization. GitHub keeps an issue's type only where the receiving organization has a matching one, and it does not say what happens to issue assignments on this kind of move.

Transferring to an organization you own needs no approval: GitHub starts the transfer when you confirm and runs it in the background, and it publishes no completion time. GitHub's preconditions:

- You have admin access to the repository.
- You can create repositories in the receiving organization.
- The organization holds no repository of the same name, and no fork in the same network (no repository forked from the same original).

A repository that is itself a single fork of a private upstream (forked on its own from a private repository) cannot be transferred at all. Old links keep working: GitHub redirects web URLs and git operations from the old path and gives no expiry date. It does not redirect a GitHub Pages site published from the repository, and creating a repository at the old path ends the redirects (see the box in Step 5).

- **Only if the repository also publishes a GitHub Pages site**

GitHub Pages is GitHub's own site hosting, separate from Cloudflare Pages; the repository's **Settings > Pages** shows whether it publishes a site there. Read on only if it publishes one on a custom domain. A private repository moving to a Free organization loses GitHub Pages, and its site is unpublished. The domain's DNS records still point at GitHub, though, so someone else can publish their own GitHub Pages site on your domain unless the domain is verified (proved to be yours in GitHub's settings). As GitHub suggests, remove or update those DNS records before you transfer a private repository, to avoid a domain takeover.

If you verified the domain for your personal account, only repositories that account owns can publish to it. The transferred repository cannot publish there until you verify the domain for the organization instead, under the organization's **Settings > Pages**.

02 Record the Pages project before touching GitHub

Capture the configuration while the project is healthy. If the repository has a Wrangler configuration file (`wrangler.toml` , `wrangler.json` or `wrangler.jsonc`), the settings it declares survive on their own: the build output directory, the compatibility settings, and any bindings and environment variables. Everything else lives only in the dashboard. Open the project from **Workers & Pages**: most of the items below are under its **Settings** tab, the custom domains are under **Custom domains**, and Web Analytics is under **Metrics**. Write down, or screenshot, each of these:

- **The build command and build output directory** (Settings > Build): the command Cloudflare runs, and the folder it publishes. If your build command does anything security-relevant, such as deleting internal files so they are never published, copy it exactly. The new project starts with no build command, and its first deployment runs whatever you enter, so a line you leave out publishes the files the old command deleted.
- **The root directory and environment variables**, for both production and preview environments. Note which variables are secret: the API below does not return their values, so take them from wherever you keep them.
- **The bindings and the compatibility date and flags**, for both environments, if the project uses Functions (server-side code in a `functions` folder or a `_worker.js` file) and has no Wrangler file. Bindings are what connect Functions to other Cloudflare services, such as a key-value store or a database (Settings > Bindings).
- **The production branch, the automatic-deployment toggles, and the preview branch rules** (which branches build previews, and any include or exclude patterns).

- **The build watch paths and whether build caching is on** (Settings > Build). Watch paths decide which changed files start a build.
- **The access policy on preview deployments**, if you require sign-in to view previews (a Cloudflare Access policy, under Settings > General).
- **Every custom domain attached to the project**, subdomains included. Projects accumulate these: a www alias, a subdomain serving a policy file, a staging name. Step 8 moves each one, and a domain you forget starts returning errors when the old project is deleted.
- **Deploy hooks** (addresses that start a deployment when something calls them), if anything calls them. Hook URLs are unique to a project, so anything calling one needs the replacement URL from the new project.
- **Web Analytics**: whether it is on, and whether it was switched on from the project's **Metrics** tab or added by hand to the HTML in your repository. Step 10 needs to know which. Check the repository rather than the live site: a project switched on from Metrics also serves the script, because Cloudflare adds it at deploy time.

You do not need to record the build system version. Build image v3 is the last one Cloudflare will release, and Cloudflare scheduled every v1 project to move onto it on 15 September 2026 and every v2 project on 23 February 2027. A build that relied on an older default language or tool version can fail on v3, so pin your versions with environment variables (such as `NODE_VERSION`). Cloudflare's move to v3 keeps them, and Step 7 has you enter them on the new project.

Optional, for developers: one API request returns most of that configuration as a file. It leaves out deploy hooks, which the Pages API does not expose; the preview access policy, which lives in Cloudflare Access; and the values of secret environment variables, which the API does not return in usable form.

Create a token under **My Profile > API Tokens** with the account-level Cloudflare Pages Read permission, put it in the `CLOUDFLARE_API_TOKEN` environment variable, and copy your account ID from **Workers & Pages > Account Details**:

```
curl -s -H "Authorization: Bearer $CLOUDFLARE_API_TOKEN" \
  "https://api.cloudflare.com/client/v4/accounts/<ACCOUNT_ID>/pages/projects/<PROJECT>" \
  > pages-project-snapshot.json
```

Open the file before you rely on it: a rejected token still writes a JSON body, so a snapshot that reads `"success": false` is an error message rather than your configuration. Delete the token once you have the snapshot. Keep `pages-project-snapshot.json` out of the repository and delete it once Step 7 is done, because it holds your environment variable names, the values of any variable not marked secret, and the project's Web Analytics token.

`npx wrangler pages download config <PROJECT>` writes a Wrangler configuration file instead. Cloudflare labels the command experimental. The file also has no field for the build command, the root directory, the production branch, the preview branch rules, the build watch paths, or build caching, so it does not replace the snapshot. Commit that file only if you mean to manage these settings from the repository. Deploying a Wrangler file that carries a `pages_build_output_dir` key makes that file the source of truth, and the dashboard then shows the fields it sets without letting you edit them.

03 Prepare the organization before the repository lands

If the organization does not exist yet, create it now. The settings below decide who can reach the repository the day it lands, so set them while the organization is empty. The CIS column gives each setting's number in the CIS GitHub Benchmark v1.2.0, the security baseline for GitHub published by the Center for Internet Security. Our guide "Set up and secure a new GitHub account and organization" (anobytes.com/resources) locks an organization down in full; this table is the subset that has to be in place before a transfer.

SETTING	CIS	SET IT TO, AND WHY
Base permissions	1.3.8	Settings > Access > Member privileges. This one setting applies to every repository in the organization. Set No permission if only specific people should see the private repository you are moving: anything above No permission lets every member, including one added later, read it. Check the current value rather than assuming one: GitHub documents a Read default for public repositories and gives no default for private ones.
Two-factor authentication	1.3.4, 1.3.5	Settings > Security > Authentication security. Require two-factor authentication (2FA: a second sign-in step beyond the password) for everyone. What happens to anyone without it: • A member keeps their membership, and still counts toward the organization's seats, but cannot reach anything in the organization until they enroll. • An outside collaborator is removed and loses repository access, and an owner can reinstate them if they enable 2FA within three months of being removed. Turning it on while the member list is one person long avoids both.
Repository creation	1.2.2	Settings > Access > Member privileges > Repository creation. Clear both boxes, Public and Private, so only owners can create repositories; a new organization lets members create them. Limiting members to private repositories needs Enterprise Cloud, GitHub's enterprise plan, so on Free and Team clearing both boxes is what stops a member creating a public one.
Plan	-	Settings > Access > Billing and licensing shows the organization's plan. On Free, a private repository can have neither branch protection rules nor rulesets, the newer form of the same control. The same limit applies on a Free personal account, so a private repository moving from one has nothing to carry across. To see whether the repository you are moving has either, look at its Settings > Branches and its Rulesets page. If it has branch protection, upgrade the organization to Team or above before the transfer, then check the rules arrived and re-create any that did not.
OAuth app access restrictions	1.4.1	Settings > Third-party Access > OAuth app policy. This covers OAuth apps, a different kind of integration from the GitHub App Cloudflare uses (Step 6). Access restrictions are on by default in a new organization; leave them on. Members and outside collaborators then need an owner's approval before an OAuth app can reach the organization's resources. A third-party OAuth app that worked with the private repository under your personal account therefore loses access to it once the repository moves, until you approve the app. GitHub's own privileged apps, the GitHub CLI among them, are exempt: members can still authorize them and use them to reach the organization's data.
GitHub Actions	-	Settings > Actions > General. If the repository's Actions tab shows no workflow runs, disable Actions for the organization, so no workflow can run with access to the repository's contents. Check the tab rather than the <code>.github/workflows</code> folder: code scanning default setup and GitHub Pages deployments run there without a workflow file. You can later enable Actions for specific repositories.

SETTING	CIS	SET IT TO, AND WHY
Personal access tokens	-	Settings > Personal access tokens > Settings. A fine-grained personal access token works for one resource owner only, so a token scoped to your personal account stops covering the repository the moment it moves; Step 5 replaces it. Leave the approval policy on its default, Require administrator approval, under which an owner approves every fine-grained token that reaches the organization except an owner's own. Classic tokens are never subject to approval and can reach the organization unless you restrict them on the Tokens (classic) tab of the same page. Restrict them if nothing you run still uses one; anything that does then loses access to the organization's private resources.

04 Transfer ownership of the repository

If you want a rollback point, take a mirror copy before you press the button:

```
git clone --mirror git@github.com:<OLD-OWNER>/<REPO>.git
```

It copies every branch, tag, and note. GitHub moves all of that intact, so the mirror is only a precaution. The mirror does not carry issues, releases, or the pull request records (titles, reviews, comments), which live only on GitHub; the pull request branches do come across, as `refs/pull/*` heads.

In the repository:

1. Open **Settings > General** and scroll to the Danger Zone.
2. Click **Transfer**, then under **New owner** choose **Select one of my organizations** and pick the organization.
3. Read the warnings. GitHub's documented steps include one about features the repository may lose on the new owner's plan, which is the protected-branches case from Step 1. As the dialog appeared in September 2026, it also warned about admin access, teams, and issue assignments. For a move into an organization you own with no teams yet, that is the permission handover from Step 1's last row. With Step 3 done, both warnings are expected.
4. Type the repository name to confirm. GitHub asks for the bare repository name, without the owner.

The command-line equivalent:

```
gh api repos/<OLD-OWNER>/<REPO>/transfer -f new_owner=<NEW-ORG>
# returns 202: the transfer runs in the background
```

When the transfer completes, open the repository at its new path and confirm that the visibility badge still says Private and that the releases page still lists your releases. CIS 1.2.6 asks you to track every change in visibility status.

05 Update every clone, script, and token

GitHub's redirect from the old path makes everything appear to keep working. Update your references anyway. The GitHub CLI has an open bug where commands run from a clone whose remote still names the old path fail silently in scripts, returning empty results instead of an error. The report covers a renamed repository, which GitHub redirects the same way it redirects a transferred one. The redirect also lasts only while the old path stays empty (see the box below).

```
git remote set-url origin git@github.com:<NEW-ORG>/<REPO>.git
git remote -v          # confirm fetch and push both show the new path
git fetch              # confirm the credentials still reach the repository
gh repo set-default <NEW-ORG>/<REPO> # only if a default was set for this clone
```

If the clone uses an HTTPS address rather than SSH, give `https://github.com/<NEW-ORG>/<REPO>.git` to the first command instead.

- Sweep for the old path anywhere it is written down: scripts calling `gh -R <old-owner>/<repo>`, other machines' clones, CI configuration, bookmarks, and documentation.
- If your SSH configuration (`~/.ssh/config`) ties a host alias to a key for the old account, point it at a key whose account can reach the organization. A key that authenticates as an account without access to the moved repository produces "Repository not found", which looks like a broken redirect and is a credentials problem.
- Replace any fine-grained personal access token that named your personal account as its resource owner. Create the new one under your **Settings > Developer settings > Personal access tokens > Fine-grained tokens**, with the organization as resource owner. Step 3 explains why the old ones stop working.
- Review what came across with the repository: its webhooks, deploy keys, and secrets survive the transfer (Step 1). Open the repository's **Settings > Webhooks**, **Deploy keys**, and **Secrets and variables** pages, delete what is no longer used, and replace (rotate) any key or secret held by someone who should not reach an organization repository. Check that each webhook you keep posts to an HTTPS URL with SSL verification enabled (CIS 1.4.4) and carries a secret.

• **Never create a new repository at the old path**

Creating a repository or a fork at the old owner and repository name permanently deletes every redirect, and with it every old link, clone URL, and script reference that still worked. For some repositories GitHub blocks the old name for good instead: it permanently retires that owner and repository name combination, and nobody can create a repository at that name again. That happens when the repository had an action listed on GitHub Marketplace, or more than 100 clones or more than 100 uses of GitHub Actions in the week before the transfer.

06 Give Cloudflare access to the organization

Cloudflare builds from GitHub through its **Cloudflare Workers and Pages** GitHub App, which has to be installed on the organization. Install it here, or let Step 7's wizard do it: the project-creation flow includes the same install-and-authorize screen. Only an organization owner can complete the install. A repository admin can normally install a GitHub App, but not one that asks for organization permissions or the repository administration permission, and Cloudflare's app asks for the latter. A member can request the installation and an owner approves it.

To install it here, open the app's own page, github.com/apps/cloudflare-workers-and-pages. Grant access to **Only select repositories** and pick the one repository. The app then holds the least access that works (CIS 1.4.3), so if the app or your Cloudflare account is compromised, its access covers only that repository.

Do not look for an install button on the organization's **Third-party Access > GitHub Apps** settings page. It lists the apps already installed there and, as it appeared in September 2026, had no button that starts a new installation. **Developer settings > GitHub Apps** is a different page again, holding the apps the organization creates itself. Cloudflare's GitHub integration page says the GitHub Apps Manager role is enough to add the installation. GitHub's installation page says the app manager role does not give a person the ability to install a GitHub App on the organization, so have an owner do it.

Cloudflare's documentation says a GitHub account should point to one Cloudflare account, and its troubleshooting page says one repository cannot be used by Pages projects in two Cloudflare accounts. Uninstalling the app revokes Cloudflare's access to every repository it covered and stops new builds for every project connected through it. Leave the installation on your personal account in place while other projects build from it.

CHOICE Decide whether to rebuild on Cloudflare Pages or Workers

You have to recreate the project either way, so decide now whether the replacement should be a Pages project. If none of the conditions in the list after the table applies, staying on Pages is the shorter path, because Steps 7 to 10 cover it in full.

Workers is Cloudflare's other way to serve a site, through its static-assets feature, and the platform Cloudflare now invests in. In April 2025 Cloudflare said that Pages will continue to be supported and that from then on all its investment, optimizations, and feature work will go to Workers. Its documentation still says to start new projects with Workers. Pages still gets changes: the most recent entry in its changelog is dated 11 August 2026. If you consider moving, read Cloudflare's migration guide in full, compatibility table included.

For a static site, these are the differences that decide it:

CAPABILITY	PAGES	WORKERS	STATIC ASSETS
_headers and _redirects files	Supported		Supported, same file format, placed in the assets directory
Excluding files from publication	A build command in the dashboard (a setting no commit can change)		A .assetsignore file in the root of the assets directory, same format as .gitignore . The list of excluded files then lives in the repository and survives any project recreation, which removes the risk in Step 2's first bullet.
Stable branch preview URLs	Yes (branch.project.pages.dev)		Yes: with preview builds on, Workers Builds (Cloudflare's build service for Workers) gives each branch a Worker Preview. Its Preview URL, <preview-name>-<worker>.<subdomain>.workers.dev , always serves the branch's latest deployment. The Preview name defaults to the branch name, and a new Worker uses Worker Previews by default.
Custom domain whose DNS is with another provider	Supported for a subdomain (such as www.example.com), by a CNAME record at the other provider. An apex domain (example.com itself) has to be a zone, a domain whose DNS Cloudflare runs, on the same Cloudflare account		Not supported: the domain has to sit in a zone whose DNS Cloudflare's nameservers serve
Preview branch include and exclude rules	Supported		Unsupported; Cloudflare's migration guide marks it as having a workaround
Default hostname	<project>.pages.dev		<worker>.<your-subdomain>.workers.dev , so rules written for a specific hostname, and bookmarks, change either way
Serving static requests	Free and unlimited		Free and unlimited, unless you turn on Workers Caching, which then bills every request, static assets included, at the standard request rate. An assets-only site needs no Worker script

Which to choose:

- Stay on Pages if your domain's DNS is with another provider, or if you rely on preview branch include and exclude rules.
- Move to Workers if your build command deletes files before publishing, since **.assetsignore** keeps that exclusion in the repository, where a project rebuild cannot lose it.
- Where both apply, stay.

Steps 7 and 8 below rebuild a Pages project. If you move to Workers instead:

- Cloudflare's migration guide covers the build settings and the custom domain.
- A Workers custom domain gets an Advanced Certificate (a dedicated certificate for that hostname), and Cloudflare says it adds no CAA records automatically for Advanced Certificates. CAA records are the DNS entries that list which certificate authorities may issue certificates for your domain. If the zone has any, confirm they allow the three authorities Step 8 names before you add the domain.
- Step 9 applies as written, and Step 10's checks apply with the **workers.dev** hostname in place of **pages.dev**.

- Take the `workers.dev` form of a hostname-specific `_headers` rule from the Workers static-assets headers page, listed in the References.
- Cloudflare documents the Metrics-tab analytics switch for Pages only, so on Workers add the site on the dashboard's Web Analytics page, unless the HTML in your repository already carries the analytics script.

07 Create the replacement Pages project

Use the dashboard for this. It is the flow Cloudflare documents, and the only one that walks you through installing and authorizing the GitHub App on the organization; the create-project API also accepts a Git source. Do not use Wrangler's `pages project create`: it takes no Git source and makes a direct-upload project, which Cloudflare's Direct Upload documentation says cannot switch to Git integration later.

Before you open the wizard:

- If your `_headers` file scopes rules to the literal old hostname, switch those rules to Cloudflare's placeholder form and commit that change now, since the first build runs as soon as you finish the wizard. A common example adds `X-Robots-Tag: noindex` to the `pages.dev` copy of a site so search engines index only the real domain. The placeholders survive any project name:

```
https://:project.pages.dev/*
X-Robots-Tag: noindex
https://:version.:project.pages.dev/*
X-Robots-Tag: noindex
```

- **Choose the new project's name.** It sets the project's `*.pages.dev` hostname, and the old project holds the old name until you delete it. Cloudflare gives no way to change a project's `pages.dev` subdomain afterwards. Its Direct Upload documentation records that a name already taken gets a few random characters appended, so pick a name different from the old one. Note the two places the old hostname may be written down: the DNS record Step 8 updates, and any hostname-specific rules in a `_headers` file.

In the wizard: go to **Workers & Pages > Create application > Pages > Connect to Git**, pick the repository under the organization, and stop at the "Set up builds and deployments" screen.

- **Enter the name you chose, and type Step 2's snapshot back in:** production branch, build command, build output directory, root directory, and environment variables, all before anything deploys. The wizard's final button is "Save and Deploy", so the first deployment already runs your build command.

Once the project exists:

- **Restore the rest of Step 2's snapshot** under the new project's Settings: any bindings and compatibility settings, the automatic-deployment toggles, the preview branch rules, the build watch paths, build caching, the access policy on preview deployments, and new deploy hooks. The hook URLs change, so update everything that calls them.
- **Then verify on the new project's own `pages.dev` address**, before the custom domain moves. Check that the site renders, that the response headers are right, and that the build log (**Deployments > View details > Build log**) echoes the command you entered, character for character. Current build logs label that line "Executing user command"; Cloudflare does not document the label. If the build command strips internal files, also request a few of their paths on this hostname and confirm each answers 404. Checking here catches a mistyped command before the custom domain points at this project.

08 Move the custom domains

Moving a domain takes it offline. It stops working when you delete its DNS record. If its DNS is with another provider, it stops when you remove it from the old project instead. It does not serve again until the new project has it and Cloudflare has issued the new certificate. Cloudflare gives no completion time, but its debugging page says to contact support if a domain is still verifying after 15 minutes, so plan for at least that, pick a quiet hour, and watch it through. Google treats DNS and network errors like server errors and slows its crawling at once, and it publishes no timeline for recovering, so keep the outage short.

The domain cannot sit on both projects while you switch: Cloudflare's dashboard refuses a domain that another project in the account already holds. Cloudflare does not document that rule; the stale-association reports in Step 9 show the same behavior. So do every check below before you delete anything, because by the time the new project refuses the domain, the old project has already stopped serving it.

Before you delete anything:

- **Clear anything that blocks the new project from taking the domain.** Cloudflare's known issues page lists three such conditions: a Cloudflare Access policy (Cloudflare's sign-in gate) already enabled on the domain, a Worker already routed on it, and a wildcard name such as `*.example.com`, which a Pages project cannot hold at all. On an Enterprise zone, where zone holds are on by default, a zone hold also stops Pages adding the domain; Cloudflare's debugging page says to release it for the setup and reinstate it afterwards. Zone holds exist only on Enterprise zones.
- **Check that nothing intercepts the validation path.** Cloudflare issues the certificate by HTTP validation: it requests a token under `http://<your-domain>/.well-known/acme-challenge/`, and the domain stays in Verifying until that request succeeds. Cloudflare names Cloudflare Access, a redirect rule, and a Worker as examples of what intercepts it. For Access, its advice is to switch it off over the domain until the domain verifies, then switch it back on. Run Cloudflare's own test below and read the status line and any `location` header. In Cloudflare's example, an intercepted path answers with a `location` header pointing at a Cloudflare Access sign-in page:

```
curl -s -o /dev/null -D - https://<your-domain>/.well-known/acme-challenge/randomstring
```

- **Check the CAA records** if your DNS is with another provider, if Universal SSL is switched off on the zone, or if the domain's parent domain has its DNS with another provider. Cloudflare's debugging page names the first two as the likely cases. CAA records are the DNS entries that list which certificate authorities may issue certificates for your domain. If Cloudflare runs the DNS for the domain and its parent, and Universal SSL (the free certificate Cloudflare issues for every zone) is on, they need nothing from you. A zone with no CAA record lets any authority issue, and where the zone holds one, Cloudflare adds its own alongside yours.
- **Where the CAA records need checking, confirm they allow the three authorities** Cloudflare issues Pages certificates from, each for `issue` and again for `issuewild` (the entry for wildcard certificates), as Cloudflare's Pages page lists them:
 - Let's Encrypt: `letsencrypt.org`
 - Google Trust Services: `pki.goog; cansignhttpexchanges=yes`
 - SSL.com: `ssl.com`

Cloudflare's CAA page says its list of authorities is not exhaustive and can change, so re-read both pages whenever you touch these records. Cloudflare says your CAA records must permit the authority it uses, and does not say it will pick another when one is blocked. A missing entry for any of the three can therefore stop the domain verifying. CAA records are inherited, so a parent domain whose DNS is with another provider has to allow the same authorities or carry none at all.

Then move the domain, detaching it in the order Cloudflare documents:

1. In the zone's DNS records (**DNS > Records**), delete the CNAME record pointing at the old project. The outage starts here.
2. In the old project's **Custom domains** tab, remove the domain.

3. In the new project, open **Custom domains > Set up a domain** and add the same domain. For a zone on the same Cloudflare account, Cloudflare writes the new CNAME record once you confirm it. Read that record back afterwards: its target must be the new project's `pages.dev` hostname.

Pages accepts a domain whose DNS is with another provider only if it is a subdomain. For such a domain, remove it from the old project, add it to the new one, then point the CNAME record at your provider to the new project's `pages.dev` hostname. On this path the outage starts when you remove the domain from the old project. Cloudflare says to add the domain in its dashboard before pointing a record at it.

- **Watch the domain's status in the new project**, since the time elapsed does not tell you whether it is active. Cloudflare's custom domains page records a second delay: a DNS record pointed away from Pages and back returns errors to visitors until it becomes active again.
- **If adding the domain to the new project fails**, re-add it to the old project, which is still intact and restores serving while you investigate.
- Repeat for every domain Step 2 listed.
- Adding the domain to the new project creates a new custom hostname (Cloudflare's entry for the domain on that project), with its own certificate. Cloudflare issues that certificate separately from the zone's own edge certificate and serves it in preference to that one, which is also why an Advanced Certificate does not work with a Pages project at all.

09 Retire the old project

Keep the old project until the new one has served real traffic on the custom domain and run at least one push-triggered deployment, the first check in Step 10. Then delete it in this order:

- Confirm every custom domain is already detached (Step 8 did this; check that the old project's Custom domains tab is empty). A project deleted while a domain is still attached can leave a stale association behind, and Cloudflare then refuses to add that domain to another project. Cloudflare does not document this; the reports come from its community forum, and clearing it takes an API call to delete the leftover record, which a community write-up in the References shows.
- **Only if the project has more than a hundred deployments:** it may refuse to delete until old deployments are removed. This command removes one at a time and asks you to confirm each; add `--force` to skip the prompt and to delete a deployment that still holds an alias (a branch address pointing at it). Cloudflare's known issues page carries a shell loop that deletes every deployment except the active production one, after which you can delete the project.

```
npx wrangler pages deployment delete <DEPLOYMENT_ID> --project-name <PROJECT>
```

- Delete the project (**Settings > Delete project**). The old `*.pages.dev` hostname stops answering, and anything still pointing at it fails from this moment; Step 7 had you note where it was written down for this reason.

10 Verify the pipeline end to end

The migration is done when every one of these passes:

- Push a small commit to the production branch, watch the deployment run through, and read its "Executing user command" line in the build log again (Step 7): it shows the build command that ran.
- `curl -I https://<your-domain>/` returns 200 with the response headers you expect, such as those your `_headers` file sets for the production domain, and the pages render in a browser.
- If your build command strips internal files, request a few of their paths on the production domain and confirm each answers 404. The build command is the only thing keeping them private, and this request proves it ran.
- The `pages.dev` copy sends the headers your hostname-specific `_headers` rules give it (for the noindex pattern: `curl -sI https://<project>.pages.dev/ | grep -i x-robots`), and the production domain does not send those.
- Branch previews still build if you use them, and anything calling a deploy hook has the new project's hook URL.

- Cloudflare Web Analytics keeps reporting if Step 2 found its script added by hand to the HTML in your repository, because that script is in the HTML the new project deploys and nothing in the old project owns it. If Step 2 found it switched on from the old project's Metrics tab instead, Cloudflare was adding the script at deploy time for that project, so switch it on again in the new one.

A Google Search Console domain property stays verified and needs no action, because its DNS record never changed.

FIX If the transfer already happened

You clicked the transfer button, and the dashboard now shows "The project is linked to a repository that no longer exists, this may cause deployments to fail". That message means the project can no longer build from Git. Cloudflare's troubleshooting page lists a separate banner that begins "This project is disconnected from your Git account", and tells you to reinstall the Git integration to clear it. That banner describes a broken installation; reinstalling repairs nothing for a transferred repository, so use one of the routes below.

- The site is still serving its last deployment; the broken link stops only new deployments.
- The old project still exists, so the settings Step 2 records are still in it. Record them now, then carry on with Step 3 and Steps 5 to 10; they work the same after the transfer as before it. Also do two checks from the steps you skipped. One is the GitHub Pages box at the end of Step 1, if the repository published a GitHub Pages site on a custom domain. The other is the visibility and releases check at the end of Step 4.
- If you need deployments back before the rebuild is ready, transfer the repository back to the original account. That is one of the two options Cloudflare's troubleshooting page gives, and afterwards the owner stored in the old project matches the repository's owner again. If the banner then reads "The repository cannot be accessed", that page says to check the app's repository access on your personal account (github.com/settings/installations) and include the repository.

REF References

Transferring a repository; what moves, redirects, and the effect of reusing the old path · docs.github.com/repositories/creating-and-managing-repositories/transferring-a-repository

Transfer a repository, REST API · docs.github.com/rest/repos/repos#transfer-a-repository

Verifying a custom domain for GitHub Pages: what verification blocks, and which repositories can publish to a verified domain · docs.github.com/pages/configuring-a-custom-domain-for-your-github-pages-site/verifying-your-custom-domain-for-github-pages

Setting base permissions for an organization · docs.github.com/organizations/managing-user-access-to-your-organizations-repositories/managing-repository-roles/setting-base-permissions-for-an-organization

Requiring two-factor authentication, and who it removes · docs.github.com/organizations/keeping-your-organization-secure/managing-two-factor-authentication-for-your-organization/requiring-two-factor-authentication-in-your-organization

Restricting repository creation, and the Enterprise Cloud limit on private-only · docs.github.com/organizations/managing-organization-settings/restricting-repository-creation-in-your-organization

OAuth app access restrictions: on by default in a new organization, and the exemption for privileged apps · docs.github.com/organizations/managing-oauth-access-to-your-organizations-data/about-oauth-app-access-restrictions

Privileged OAuth apps, GitHub CLI included · docs.github.com/apps/oauth-apps/using-oauth-apps/privileged-oauth-apps

Setting a personal access token policy for an organization: the approval default, and classic tokens · docs.github.com/organizations/managing-programmatic-access-to-your-organization/setting-a-personal-access-token-policy-for-your-organization

Installing a GitHub App: where the install starts, and who may run it · docs.github.com/apps/using-github-apps/installing-a-github-app-from-a-third-party

GitHub CLI: silent failures in scripts against a renamed repository's remote, open issue · github.com/cli/cli/issues/11754

CIS GitHub Benchmark v1.2.0, 18 February 2026 · cisecurity.org/cis-benchmarks

Cloudflare Pages GitHub App: organizational access, and what uninstalling removes · developers.cloudflare.com/pages/configuration/git-integration/github-integration

Cloudflare Pages dashboard banners, including the one a transferred repository produces · developers.cloudflare.com/pages/configuration/git-integration/troubleshooting

Cloudflare API schema (openapi.json): the calls that disconnect and connect an existing Pages project's Git source, absent from the API reference · github.com/cloudflare/api-schemas

Cloudflare changelog: the Cloudflare CLI in beta, 28 September 2026 · developers.cloudflare.com/changelog/post/2026-09-28-cloudflare-cli-beta

Cloudflare CLI source, including its pages source connect and disconnect commands · github.com/cloudflare/cf

Cloudflare Pages build image: the version policy and the v1 and v2 dates · developers.cloudflare.com/pages/configuration/build-image

Cloudflare Pages configuration in a Wrangler file, and downloading a project's settings into one · developers.cloudflare.com/pages/functions/wrangler-configuration

Wrangler's Pages commands, including the experimental label on download config · developers.cloudflare.com/workers/wrangler/commands/pages

Cloudflare Pages custom domains: adding one, deleting one, the CAA record list, and repointing DNS · developers.cloudflare.com/pages/configuration/custom-domains

Cloudflare Pages headers file, including the pages.dev placeholder pattern · developers.cloudflare.com/pages/configuration/headers

Cloudflare Pages debugging: HTTP validation for a custom domain, what blocks it, zone holds, and when to contact support · developers.cloudflare.com/pages/configuration/debugging-pages

Cloudflare zone holds, on by default for Enterprise zones · developers.cloudflare.com/fundamentals/account/account-security/zone-holds

Cloudflare Pages known issues: the pages.dev subdomain cannot be changed, what blocks a custom domain, and deleting a project with many deployments · developers.cloudflare.com/pages/platform/known-issues

Cloudflare Pages Direct Upload: no later switch to Git integration, and the random suffix on a project name already taken · developers.cloudflare.com/pages/get-started/direct-upload

Migrate from Pages to Workers, with the compatibility matrix · developers.cloudflare.com/workers/static-assets/migration-guides/migrate-from-pages

Worker Previews: the Preview URL format for each branch · developers.cloudflare.com/workers/previews

Worker Previews setup: the Preview name defaults to the Git branch · developers.cloudflare.com/workers/previews/get-started

Workers Builds preview builds, and Worker Previews as the default for a new Worker · developers.cloudflare.com/workers/ci-cd/builds/build-branches

Workers static assets: the assetsignore file · developers.cloudflare.com/workers/static-assets/binding

Workers static assets headers file, including the workers.dev host form · developers.cloudflare.com/workers/static-assets/headers

Workers pricing: requests to static assets, and what Workers Caching bills · developers.cloudflare.com/workers/platform/pricing

Workers Cache: every request billed at the standard rate once caching is on, static assets included · developers.cloudflare.com/workers/cache

Workers custom domains: the Advanced Certificate a custom domain creates · developers.cloudflare.com/workers/configuration/routing/custom-domains

Pages Functions pricing: requests to static assets are free and unlimited · developers.cloudflare.com/pages/functions/pricing

Wrangler configuration: the main key is optional for an assets-only Worker · developers.cloudflare.com/workers/wrangler/configuration

Cloudflare for SaaS certificate and hostname priority · developers.cloudflare.com/ssl/reference/certificate-and-hostname-priority

A custom domain left associated with a deleted Pages project, community reports rather than Cloudflare documentation · community.cloudflare.com/t/release-custom-domain-from-stale-association/932086

Clearing that leftover record with an API call, a community write-up rather than Cloudflare documentation · joshtronic.com/2025/06/29/reusing-deleted-custom-domain-cloudflare-pages

Cloudflare's certificate authorities and the CAA record value for each · developers.cloudflare.com/ssl/reference/certificate-authorities

Adding CAA records: when Cloudflare writes them for you, none for Advanced Certificates, that the list is not exhaustive, and how a subdomain inherits its parent's records · developers.cloudflare.com/ssl/edge-certificates/caa-records

Cloudflare Pages changelog · developers.cloudflare.com/changelog/product/pages

Cloudflare GitHub App's requested permissions, including repository administration, 19 December 2024 ·

developers.cloudflare.com/pages/platform/changelog

Cloudflare on starting new projects with Workers · developers.cloudflare.com/pages

Cloudflare on Workers as the platform for new projects, 8 April 2025 · blog.cloudflare.com/full-stack-development-on-cloudflare-workers

Workers Builds API reference: creating a deploy hook, which the Pages API cannot do ·

developers.cloudflare.com/api/resources/workers_builds/subresources/deploy_hooks

Cloudflare Pages Web Analytics: enabling it from the project's Metrics tab, and snippet injection at deploy time ·

developers.cloudflare.com/pages/how-to/web-analytics

Cloudflare Web Analytics: adding a site on the Web Analytics page · developers.cloudflare.com/web-analytics/get-started

Google on DNS and network errors: treated like server errors, and crawling slows at once ·

developers.google.com/crawling/docs/troubleshooting/dns-network-errors