anoBytes | TECHNOLOGY SOLUTIONS

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, the project settings to record 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 August 2026. Published 2026-08-31.

10 steps, snapshot to verification

the transfer itself finishes in minutes; the Pages rebuild is the part that needs planning

2 documented recovery options once the link breaks

transfer the repository back, or build a new Pages project; the dashboard has no reconnect option

SCOPE

What this guide covers, and the two facts that shape it

Moving a repository from a personal GitHub account into an organization is a routine step as a business grows: the code stops belonging to one person's account and starts belonging to the company. When Cloudflare Pages builds and serves a site from that repository, the move takes planning, because the Pages project keeps a reference to the repository's old owner and offers no way to update it. This guide walks that move end to end. An organization-to-organization transfer breaks the same connection in the same way, so the steps apply to it unchanged. The claims below come from both platforms' documentation and observed behavior in August 2026.

Your live site stays up throughout

Cloudflare serves your site from deployments already uploaded to its network. The Git connection only triggers new builds. When the transfer breaks that connection, the dashboard warns that deployments may fail; visitors keep getting the last deployment on every domain the project serves. Nothing in this migration takes the site down except the custom-domain move in Step 8, and that window is brief.

The Pages project will not follow the repository

Cloudflare's documentation gives exactly two options for a transferred repository: transfer it back, or create a new Pages project at the new location. The GitHub App that grants Cloudflare access is installed per 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; its stored reference to the old owner stays stale.

Two steps come before the transfer, and both protect you

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.

01 Know what the transfer moves, and what it leaves behind

GitHub's transfer carries almost everything with the repository. The exceptions are the ones that catch people.

WHAT	AFTER THE TRANSFER
Git data, releases, tags	All move. Commit history, contributions, annotated tags, and the releases built on them arrive intact.
Issues, pull requests, wiki, stars, watchers	All move. Issue assignments to people outside the new organization are cleared; assignments to members survive.
Webhooks, deploy keys, secrets	Remain associated with the repository. The Cloudflare webhook survives too; it just points at a project that no longer recognizes the repository, which is why a surviving webhook does not save the connection.
Visibility	A private repository stays private. One edge case, straight from GitHub's transfer documentation: a private repository transferred to a free account or organization loses paid-plan features such as protected branches. Check what protections you had before you move.
GitHub App installations	Stay behind. Apps are installed on an account or organization, and access is granted per installation, so every app that could reach the repository under your account loses it. Step 6 reinstalls the Cloudflare app on the organization.
Permissions	The organization's default repository permissions and member privileges apply the moment the repository arrives, which is why Step 3 sets them first. GitHub also adds the previous owner as a collaborator on the transferred repository; as the organization's owner you can remove that entry, since owner access already covers you.

Transferring to an organization you own completes without an approval step: GitHub starts the transfer when you confirm, runs it in the background, and finishes within minutes for most repositories. The only precondition is that the organization has no repository with the same name. Old links keep working: GitHub redirects web URLs and git operations from the old path and documents no expiry. The one action that deletes the redirects permanently is creating a repository at the old path, which Step 5 covers.

02 Record the Pages project before touching GitHub

A Pages project's configuration lives in Cloudflare's dashboard, and none of it is in your repository. The old project is the only record of these values, so capture them while it is healthy. Open the project and write down, or screenshot, each of these:

- **The build command and build output directory** (Settings > Builds & deployments). 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, its first deploy runs whatever you enter, and a missing line publishes the files the old command deleted.
- **The root directory and environment variables**, for both production and preview environments.
- **The production branch, the automatic-deployment toggles, and the preview branch rules** (which branches build previews, and any include or exclude patterns).
- **The access policy on preview deployments**, if you require sign-in to view previews.
- **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 stops resolving when the old project is deleted.
- **Deploy hooks**, if anything calls them. Hook URLs are unique to a project, so anything calling one needs the replacement URL from the new project.

The API returns the whole configuration in one request, which makes a faithful snapshot easy to keep. The token is created in the dashboard under **My Profile > API Tokens** and needs only the account-level Cloudflare Pages Read permission:

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

03 Prepare the organization before the repository lands

If the organization does not exist yet, create it now. The organization's defaults apply to the transferred repository from the moment it arrives, so set them while the organization is empty. The settings below decide who can reach the repository the day it lands; the CIS GitHub Benchmark v1.2.0 reference for each is in the second column. Our guide "Set up and secure a new GitHub account and organization" walks the creation and the full lockdown; this table is the subset that has to precede a transfer.

SETTING	CIS	SET IT TO, AND WHY
Base permissions	1.3.8	**Settings > Access > Member privileges**. Set **No permission** for a private repository that only specific people should see. GitHub's default is Read, and the transfer documentation says these defaults apply to the transferred repository, so a member added later would otherwise read it automatically.
Two-factor authentication	1.3.4, 1.3.5	**Settings > Authentication security**. Require two-factor authentication (2FA: a second sign-in step beyond the password) for everyone. Turning it on later removes outside collaborators who lack it, so it is cleanest while the member list is one person long.
Repository creation	1.2.2	**Member privileges**. Leave creation with owners, and set the default visibility for new repositories to private. Neither changes the transferred repository; both prevent the accident where someone recreates something public later.
OAuth app access restrictions	1.4.1	On by default in a new organization; leave them on. One consequence to know about: the GitHub CLI authenticates as an OAuth app, so until you approve it for the organization (or grant it organization access when it next prompts), **gh** commands against the private repository can return 404 while plain git over SSH works. If that pair of symptoms appears after the transfer, this policy is where to look.
GitHub Actions	-	**Settings > Actions**. If the repository runs no workflows, disable Actions for the organization. A workflow that never runs cannot leak the repository's contents, and you can re-enable per repository the day you need it.
Personal access tokens	-	Fine-grained personal access tokens are scoped to one resource owner, so any token scoped to your personal account stops covering the repository the moment it moves. Plan to create replacements under the organization, and set the organization's token-approval policy while you are on the screen.

04 Transfer the repository

In the repository: **Settings > General**, scroll to the Danger Zone, choose **Transfer ownership**, pick the organization, and type the repository path to confirm. GitHub warns about admin access, teams, and issue assignments on the confirmation screen; for a move into an organization you own with no teams yet, those warnings describe the permission handover from Step 1's last row and nothing more. The command-line equivalent:

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

When it completes, open the repository at its new path and confirm two things before moving on: the visibility badge still says Private, and the releases page still lists your releases.

05 Update every clone, script, and token

GitHub redirects the old path for as long as it stays empty, so everything appears to keep working. Update your references anyway, for two reasons: the GitHub CLI has an open bug where commands against a moved remote fail silently in scripts, returning empty results instead of an error; and the redirect survives only until someone creates a repository at the old path.

```
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
```

- 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 pins a key to a host alias for the old account, update it. 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.
- Reissue any fine-grained personal access token that named your personal account as resource owner (Step 3 explains why they stop working).

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

  The redirects from the old path to the new one live only while the old path stays empty. Creating a repository or accepting a fork at the old owner-and-name permanently deletes every redirect, and with it every old link, clone URL, and script reference that still worked. GitHub offers no way to restore them.

06 Give Cloudflare access to the organization

Install the **Cloudflare Workers and Pages** GitHub App on the organization: in GitHub, **Organization settings > GitHub Apps**, or let the Cloudflare wizard in Step 7 walk you through it, since the project-creation flow includes the install-and-authorize screen. Only an organization owner can install it (a member can request the installation for an owner to approve; GitHub's App Manager role does not grant installation). Grant access to **Only select repositories** and pick the one repository; the app then holds the least access that works (CIS 1.4.3), and a compromise of either side exposes one repository instead of all of them.

Cloudflare's documentation adds one warning worth repeating: a GitHub account or organization should be connected to one Cloudflare account, and uninstalling the app revokes Cloudflare's access to every repository it covered, so an uninstall affects every project connected through it.

07 Create the replacement Pages project

Use the dashboard for this. The API accepts a Git source when creating a project, but wiring a working connection that way is unreliable in practice, wrangler creates only direct-upload projects, and Cloudflare's own maintainers point people to the dashboard for Git-connected projects. In the Cloudflare dashboard: **Workers & Pages > Create > Pages > Connect to Git**, pick the repository under the organization, and stop at the "Set up builds and deployments" screen.

- **This screen is where Step 2's snapshot gets typed 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. There is no window where the project deploys without it.
- **The rest of Step 2's snapshot is restored after creation**, under the new project's Settings: the preview branch rules, the access policy on preview deployments, and new deploy hooks, whose URLs replace the old ones wherever they are called.

- **Pick the project name knowing it sets the `*.pages.dev` hostname**, and that the old project still holds the old name. You can reuse the old name only after the old project is deleted. Even then, deleted names sometimes fail to recreate (Cloudflare's community threads document an opaque error, fixed only by support), and a freed name is claimable by any Cloudflare account in the meantime. The dependable path is a new name and a plan for the two places the old hostname may be written down: the DNS record Step 8 updates, and any host-scoped rules in a `_headers` file.
- **If your `_headers` file scopes rules to the literal old hostname** (a common pattern adds `X-Robots-Tag: noindex` to the `pages.dev` copy of a site so search engines index only the real domain), switch those rules to Cloudflare's placeholder form before the move, and they survive any project name:

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

- **Then verify on the new project's own `pages.dev` URL** before the custom domain moves: the site renders, the response headers are right, and the deploy log's "Executing user command" line shows the build command you entered, character for character. If the build command strips internal files, also request a few of their paths on this hostname and confirm each answers 404, while the project still serves no production traffic. Step 10 repeats that check on the production domain, and this is the run that catches a mistyped command at no cost.

08 Move the custom domains

A custom domain attaches to one Pages project at a time, so the domain cannot sit on both projects while you switch. The move is two actions: remove the domain from the old project, then add it to the new one, and the domain returns errors between the two. Both are quick dashboard actions in the same account, so doing them back to back keeps the window to about a minute; pick a quiet hour anyway.

- In the old project: **Custom domains**, remove the domain. In the new project: **Custom domains > Set up a domain**, add the same domain. For a zone on the same Cloudflare account, Cloudflare adds the CNAME record (the DNS entry that maps your domain onto the project's `pages.dev` address) once you confirm it. The documentation describes adding a record and says nothing about repointing one that already exists, so finish by checking the record in the zone's DNS: its target must be the new project's `pages.dev` hostname. 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. An alias or subdomain you forget keeps serving until the old project is deleted and fails only then, which is why the deletion waits for Step 9.
- The zone's existing edge certificate keeps covering the domain, so a same-account move needs no certificate wait. If you maintain CAA records by hand (the DNS entries that list which certificate authorities may issue for your domain), confirm they allow the authorities Cloudflare issues from in 2026: Let's Encrypt (`letsencrypt.org`), Google Trust Services (`pki.goog`), and SSL.com (`ssl.com`). A missing entry blocks the next certificate renewal rather than today's serving, which makes it the kind of failure that surfaces weeks later.

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. 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 that blocks the domain from ever being added to another project, and the cleanup is an API call to delete the leftover record. The check takes seconds.
- A project with more than about a hundred deployments may refuse to delete until old deployments are removed; `wrangler pages deployment delete` clears them.
- Delete the project (**Settings > General > Delete project**). The old `*.pages.dev` hostname stops resolving; anything still pointing at it fails from this moment, which is why Step 7 planned for the places it was written down.

10 Verify the pipeline end to end

The migration is done when every one of these passes, and together they take about ten minutes:

- Push a small commit to the production branch and watch the deployment run through. In the deploy log, read the "Executing user command" line once more; it shows the command that actually ran.
- `curl -I https://<your-domain>/` returns 200 with the response headers you expect, and the pages render in a browser.
- If internal files are stripped at build time, 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 is the request that proves it ran.
- The `pages.dev` copy sends the headers your `_headers` rules promise it (for the noindex pattern: `curl -sI https://<project>.pages.dev/ | grep -i x-robots`), and the production domain does not send them.
- Branch previews still build if you use them, and anything calling a deploy hook has the new project's hook URL.
- Two things need no action, and knowing so saves a false alarm: Cloudflare Web Analytics keeps reporting, because its snippet token belongs to the account and hostname rather than the project; and a Google Search Console domain property stays verified, because its DNS record never changed. Expect a brief dip in crawl rate after any hosting change; it recovers on its own.

CHOICE Rebuilding anyway? Weigh Workers static assets first

Since you have to recreate the project regardless, this is the natural moment to ask whether the replacement should be a Pages project at all. As of August 2026, Cloudflare Pages remains fully supported, and Cloudflare's documentation says plainly to start new projects on Workers, where all feature investment now goes; the Pages changelog's last entry is from April 2025. Workers serves static sites through its static-assets feature, and the migration guide's compatibility table is worth reading in full. The rows that decide it for a static site:

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 repository, same format as `.gitignore`. The exclusion list becomes version-controlled and survives any project recreation, which removes the risk Step 2 exists to guard against.
Stable branch preview URLs	Yes (`branch.project.pages.dev`)	Listed as coming soon; previews today get per-version hashed URLs
Default hostname	`<project>.pages.dev`	`<worker>.<account>.workers.dev`, so host-scoped rules and bookmarks change either way
Serving static requests	Free	Free and unlimited; a Worker script is optional for an assets-only site

A working rule: a site that depends on stable branch-preview hostnames stays on Pages for now, and you revisit when the aliases ship; a site whose deploy depends on a dashboard-only exclusion command gains real safety from moving the exclusion into the repository with `.assetsignore`.

FIX If the transfer already happened

You clicked the transfer button, and the dashboard now shows "This project is disconnected from your Git account". Three facts limit the damage:

- The site is still serving. The broken link stops new deployments; it does not touch the deployed copy.

- The old project still exists, so Step 2's snapshot is still available. Capture it now, then continue from Step 3; every remaining step works the same after the transfer as before it.
- If you need deployments back before the rebuild is ready, transferring the repository back to the original account restores the old project's connection. That is the one documented rollback, and it works because the project's stored reference to the old owner becomes correct again.

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`

Base permissions, two-factor requirement, OAuth app access restrictions, token policy · `docs.github.com/organizations`

GitHub CLI: silent failures against a moved remote, open issue · `github.com/cli/cli/issues/11754`

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

Cloudflare Pages Git integration, GitHub App installation, and troubleshooting a transferred repository · `developers.cloudflare.com/pages/configuration/git-integration`

Cloudflare Pages custom domains · `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 known issues: pages.dev names, deletion limits, Git and direct-upload projects · `developers.cloudflare.com/pages/platform/known-issues`

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

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

Cloudflare's certificate authorities · `developers.cloudflare.com/ssl/reference/certificate-authorities`

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