Cloud-Based Deployment Platform

Proof of Concept Documentation

Manuel Hanifl

MSE-BB-3-WS2025-CPS

February 4, 2026

Contents

1 Introduction

This project is a proof of concept for a cloud-based deployment platform using Cloudflare. The system provides a central dashboard to deploy applications to different environments. Deployments are triggered via cloud provider APIs and represent real deployment actions in a simplified form.

2 Architecture

The system consists of four main parts:

1. **Deployment Dashboard:** displays all the apps, their status and environment
2. **Deployment API:** serverless backend worker that exposes an API used by the dashboard. It validates requests, enforces authentication, triggers deployments and retrieves metadata
3. **Deployable Applications:** in this case just two static HTML pages app-a and app-b
4. **Deployment Log:** key-value store for deployment history

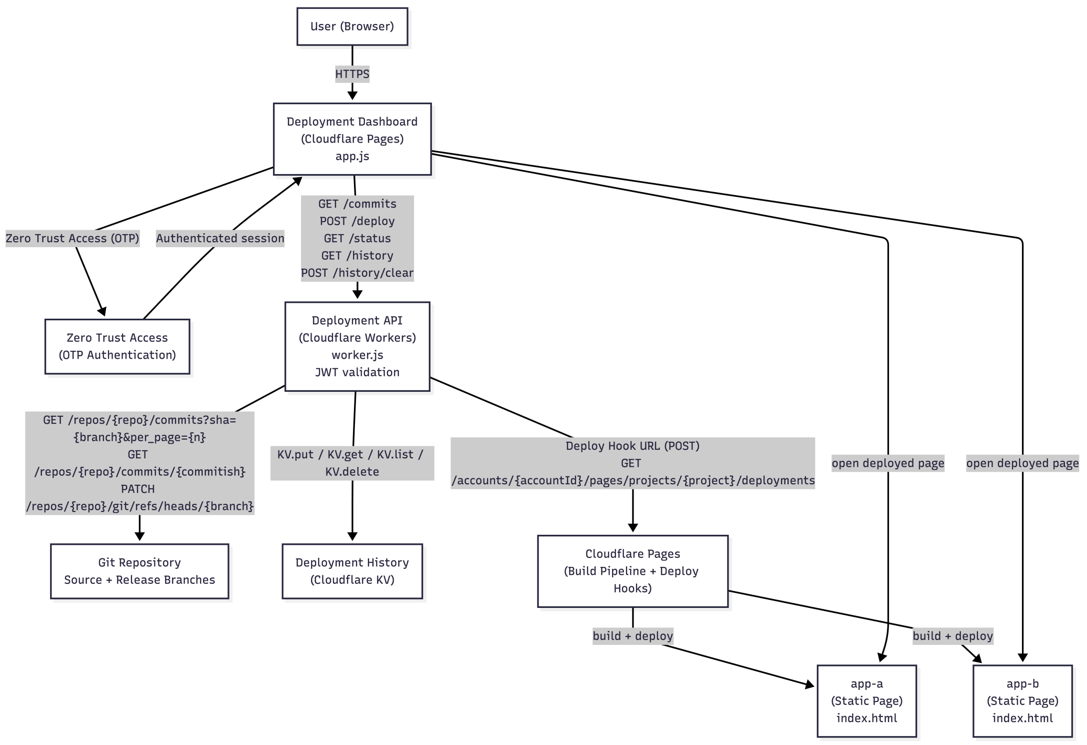

User (Browser)
HTTPS
Deployment Dashboard (Cloudflare Pages) app.js
Zero Trust Access (OTP)
Authenticated session
GET /commits
POST /deploy
GET /status
GET /history
POST /history/clear
Zero Trust Access (OTP Authentication)
Deployment API (Cloudflare Workers) worker.js JWT validation
GET /repos/{repo}/commits?sha={branch}&per_page={n}
GET /repos/{repo}/commits/{commitish}
PATCH /repos/{repo}/git/refs/heads/{branch}
KV.put / KV.get / KV.list / KV.delete
Deploy Hook URL (POST)
GET /accounts/{accountId}/pages/projects/{project}/deployments
open deployed page
open deployed page
Git Repository Source + Release Branches
Deployment History (Cloudflare KV)
Cloudflare Pages (Build Pipeline + Deploy Hooks)
build + deploy
build + deploy
app-a (Static Page) index.html
app-b (Static Page) index.html

Figure 1: Architecture Diagram

3 Deployment

All deployment and hosting is handled using Cloudflare services.

- The deployment dashboard and the deployable applications are hosted as Cloudflare Pages.

- The deployment backend is implemented and deployed as a Cloudflare Worker.
- All code is stored in a GitHub repository and connected to the deployment platform.
- Authentication is enforced using Cloudflare Zero Trust with one-time password (OTP) login.

Two environments are supported:

- **dev**: source branch `dev`, deployed via the release branch `release/app-*-dev`
- **prod**: source branch `main`, deployed via the release branch `release/app-*-main`

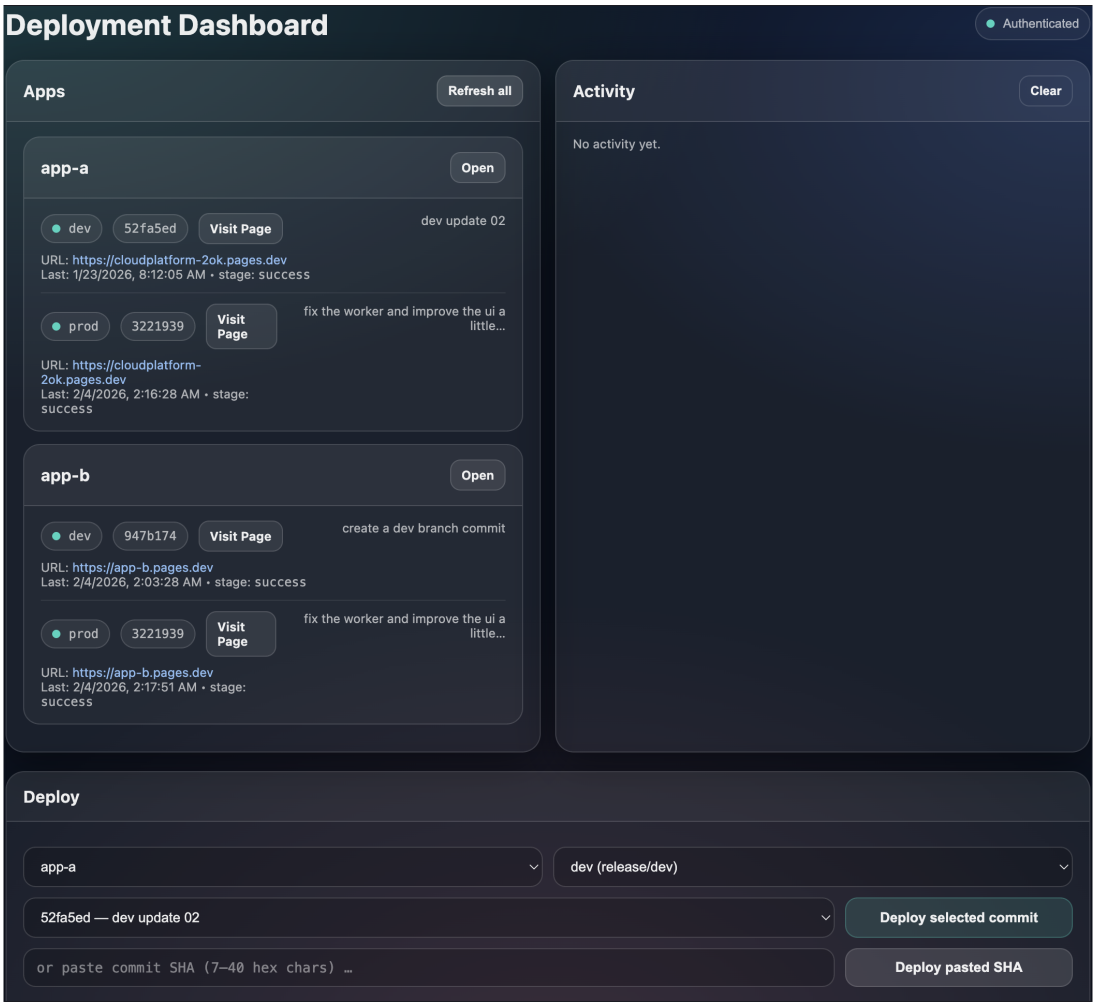
Deployments are not triggered directly by Git pushes. Instead, the deployment API explicitly updates the corresponding release branch to a selected commit and then triggers a Cloudflare Pages deploy hook. This decouples development branches from deployments.

3.1 Deployment Process

When a deployment is triggered from the frontend, the user selects a commit, an app and a target environment. The API validates the request and then updates the correct release branch to point to the selected commit using the GitHub API. For example: The user selects `{commit: 947b174, app: app-a, env: dev}`. The api updates the branch `release/app-a-dev` to point at `947b174`. After updating the release branch, the Deployment API triggers a deployment hook on the corresponding Cloudflare Page and writes a log entry in the KV log. The Cloudflare Page then builds and deploys that commit.

3.2 Usage Flow

The user opens the deployment dashboard and has an overview over all deployable apps, their current deployed commit and environment.



No activity yet.

● dev 52fa5ed Visit Page dev update 02
URL: https://cloudplatform-2ok.pages.dev
Last: 1/23/2026, 8:12:05 AM • stage: success

● prod 3221939 Visit Page fix the worker and improve the ui a little...
URL: https://cloudplatform-2ok.pages.dev
Last: 2/4/2026, 2:16:28 AM • stage: success

● dev 947b174 Visit Page create a dev branch commit
URL: https://app-b.pages.dev
Last: 2/4/2026, 2:03:28 AM • stage: success

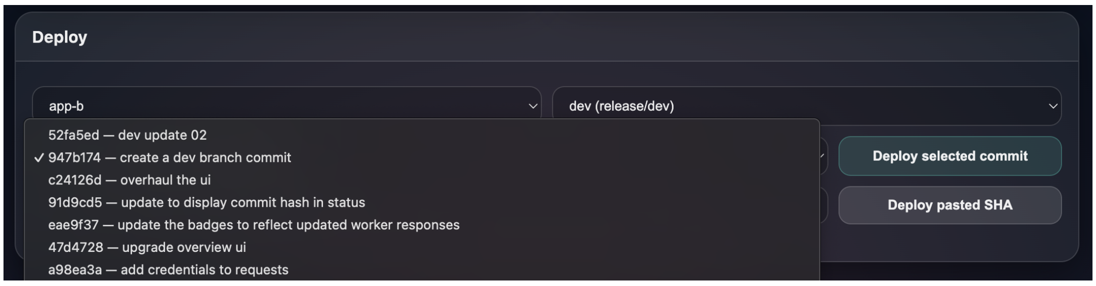
● prod 3221939 Visit Page fix the worker and improve the ui a little...
URL: https://app-b.pages.dev
Last: 2/4/2026, 2:17:51 AM • stage: success

app-a
dev (release/dev)
52fa5ed — dev update 02

or paste commit SHA (7–40 hex chars) …

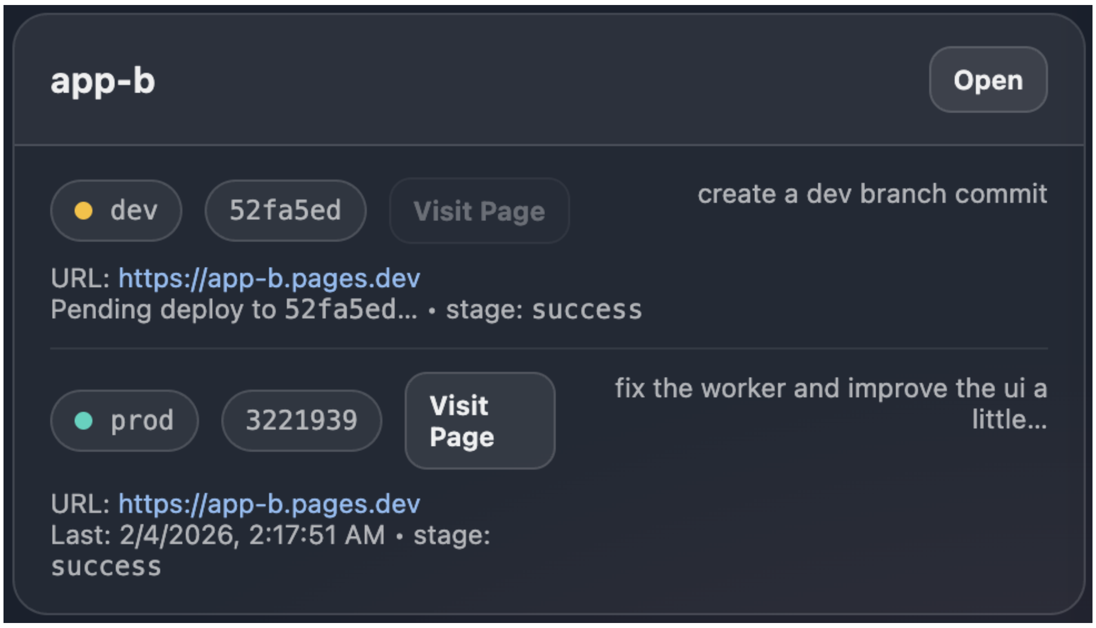
Figure 2: Deployment Dashboard

The user selects an app, an environment and a commit to deploy. The user then clicks the deploy button and the deployment process starts.



app-b
dev (release/dev)
52fa5ed — dev update 02
✓ 947b174 — create a dev branch commit
c24126d — overhaul the ui
91d9cd5 — update to display commit hash in status
eae9f37 — update the badges to reflect updated worker responses
47d4728 — upgrade overview ui
a98ea3a — add credentials to requests

Figure 3: Select Commit


app-b
Open

● dev · 52fa5ed · Visit Page · create a dev branch commit

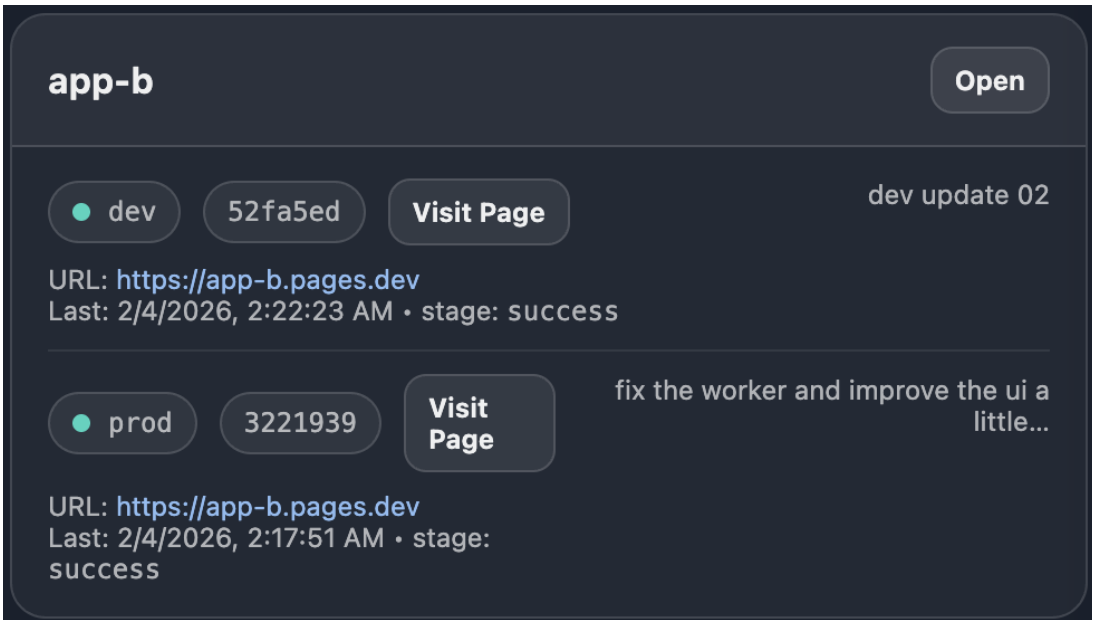
URL: https://app-b.pages.dev
Pending deploy to 52fa5ed... • stage: success

● prod · 3221939 · Visit Page · fix the worker and improve the ui a little...

URL: https://app-b.pages.dev
Last: 2/4/2026, 2:17:51 AM • stage: success

Figure 4: Deployment in Progress

The app then goes into a pending state until deployment is finished.


app-b
Open

● dev · 52fa5ed · Visit Page · dev update 02

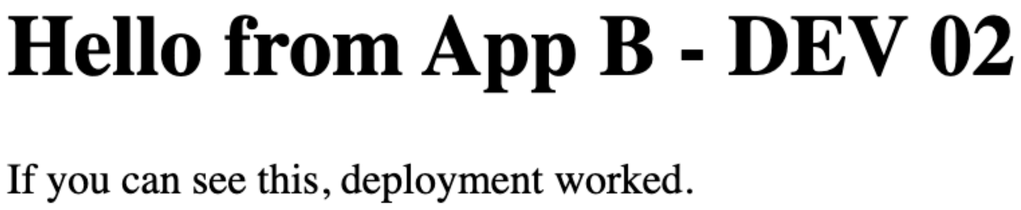
URL: https://app-b.pages.dev
Last: 2/4/2026, 2:22:23 AM • stage: success

● prod · 3221939 · Visit Page · fix the worker and improve the ui a little...

URL: https://app-b.pages.dev
Last: 2/4/2026, 2:17:51 AM • stage: success

Figure 5: Finished Deployment

Once the deployment is finished, the new commit and environment are shown in the dashboard and the user can click on the app to open it.

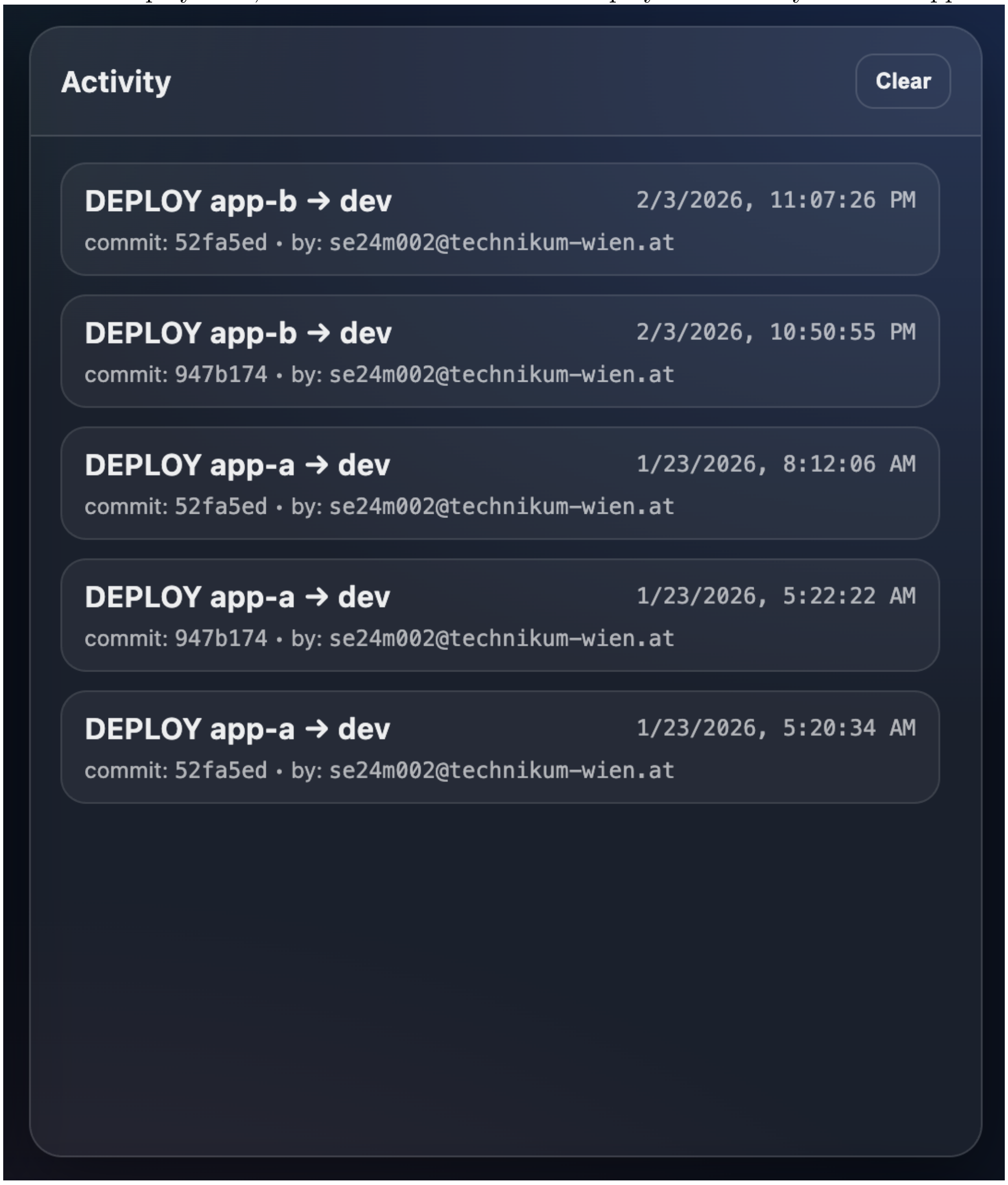


Hello from App B - DEV 02

If you can see this, deployment worked.

Figure 6: Deployed App

After deployment, the user can also view the deployment history for each app.



```
DEPLOY app-b → dev                                2/3/2026, 11:07:26 PM
commit: 52fa5ed • by: se24m002@technikum-wien.at

DEPLOY app-b → dev                                2/3/2026, 10:50:55 PM
commit: 947b174 • by: se24m002@technikum-wien.at

DEPLOY app-a → dev                                1/23/2026, 8:12:06 AM
commit: 52fa5ed • by: se24m002@technikum-wien.at

DEPLOY app-a → dev                                1/23/2026, 5:22:22 AM
commit: 947b174 • by: se24m002@technikum-wien.at

DEPLOY app-a → dev                                1/23/2026, 5:20:34 AM
commit: 52fa5ed • by: se24m002@technikum-wien.at
```

Figure 7: Deployment History

4 Cloud Services

This project is implemented using a single cloud provider: Cloudflare. Multiple cloud services are combined to implement hosting, deployment, backend logic, and security.

4.1 Reason for Selection

The initial plan was to implement the proof of concept using AWS. However, the initial setup effort and account onboarding/setup were time-consuming and accessing the free tier proved cumbersome. Therefore Cloudflare was chosen as an alternative because of its simpler entry point while still offering all the required features and services. It allows fast setup, has a strong free tier and provides deployment mechanisms such as preview environments and deploy hooks.

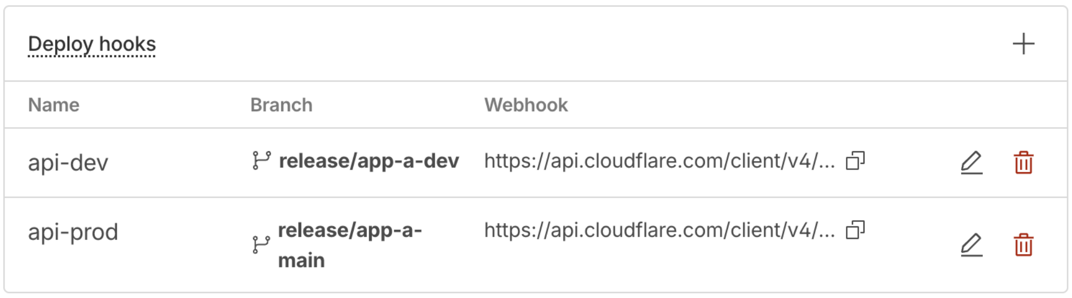
4.2 Cost Structure

This proof of concept was implemented entirely within the free tiers of the used cloud services.

That being said, if this were to scale into an actual project here would be the rough steps regarding costs of the different services:

1. **Cloudflare Pages:** Cloudflare Pages allows up to 500 builds per month and enforces a maximum size of 25 MiB per uploaded asset[1].
2. **Cloudflare Workers:** The free tier includes up to 100,000 requests per day. Beyond this limit, usage is billed per request (approximately 0.30 € per million requests)[2].
3. **Cloudflare Zero Trust Access:** Zero Trust Access is free for up to 50 users. Costs scale per user beyond that point (approximately 7 € per user per month)[3].
4. **Cloudflare Workers KV:** free tier includes 1 GB storage and daily read/write/delete allowances; beyond free usage, reads are 0.50 € per million, writes/deletes 5.00 € per million, and storage 0.50 € per GB-month[4].

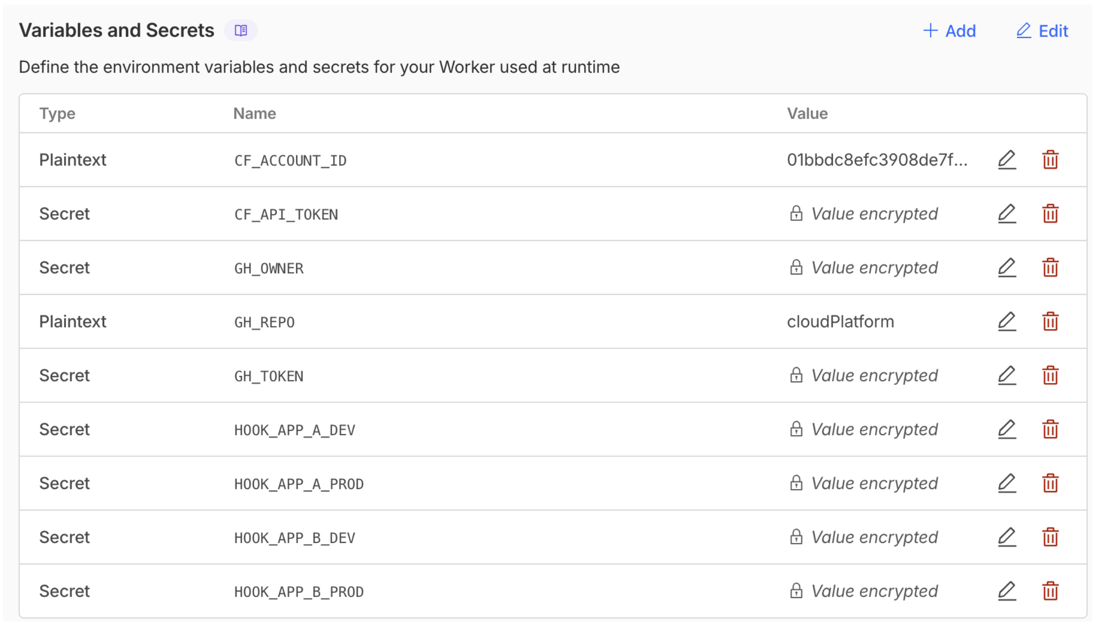
4.3 Configuration of Cloud Services


Deploy hooks

Name	Branch	Webhook
api-dev	release/app-a-dev	https://api.cloudflare.com/client/v4/...
api-prod	release/app-a-main	https://api.cloudflare.com/client/v4/...

Figure 8: Cloudflare Pages deploy hook configuration

The pages are configured to deploy from the corresponding release branches for each app and environment.



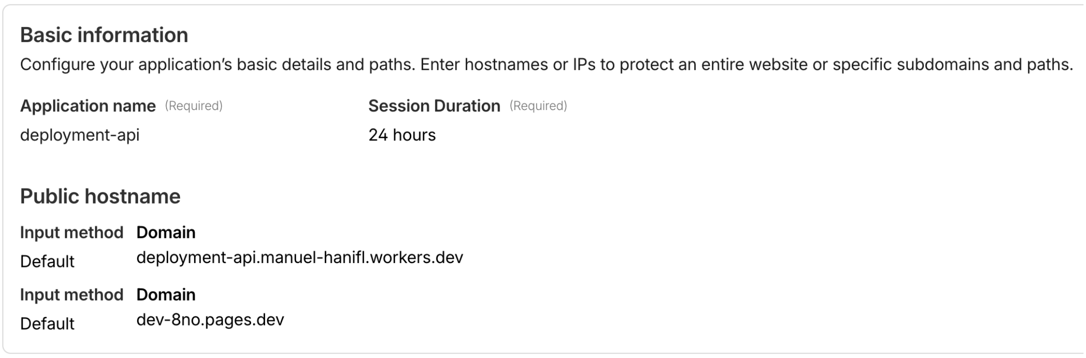
Variables and Secrets

Define the environment variables and secrets for your Worker used at runtime

Type	Name	Value
Plaintext	`CF_ACCOUNT_ID`	01bbdc8efc3908de7f...
Secret	`CF_API_TOKEN`	*Value encrypted*
Secret	`GH_OWNER`	*Value encrypted*
Plaintext	`GH_REPO`	cloudPlatform
Secret	`GH_TOKEN`	*Value encrypted*
Secret	`HOOK_APP_A_DEV`	*Value encrypted*
Secret	`HOOK_APP_A_PROD`	*Value encrypted*
Secret	`HOOK_APP_B_DEV`	*Value encrypted*
Secret	`HOOK_APP_B_PROD`	*Value encrypted*

Figure 9: Cloudflare Worker API variables

Sensitive values such as API tokens and deploy hook URLs are stored as Worker environment variables to avoid exposing them in the code.



Basic information

Configure your application's basic details and paths. Enter hostnames or IPs to protect an entire website or specific subdomains and paths.

Application name (Required): deployment-api

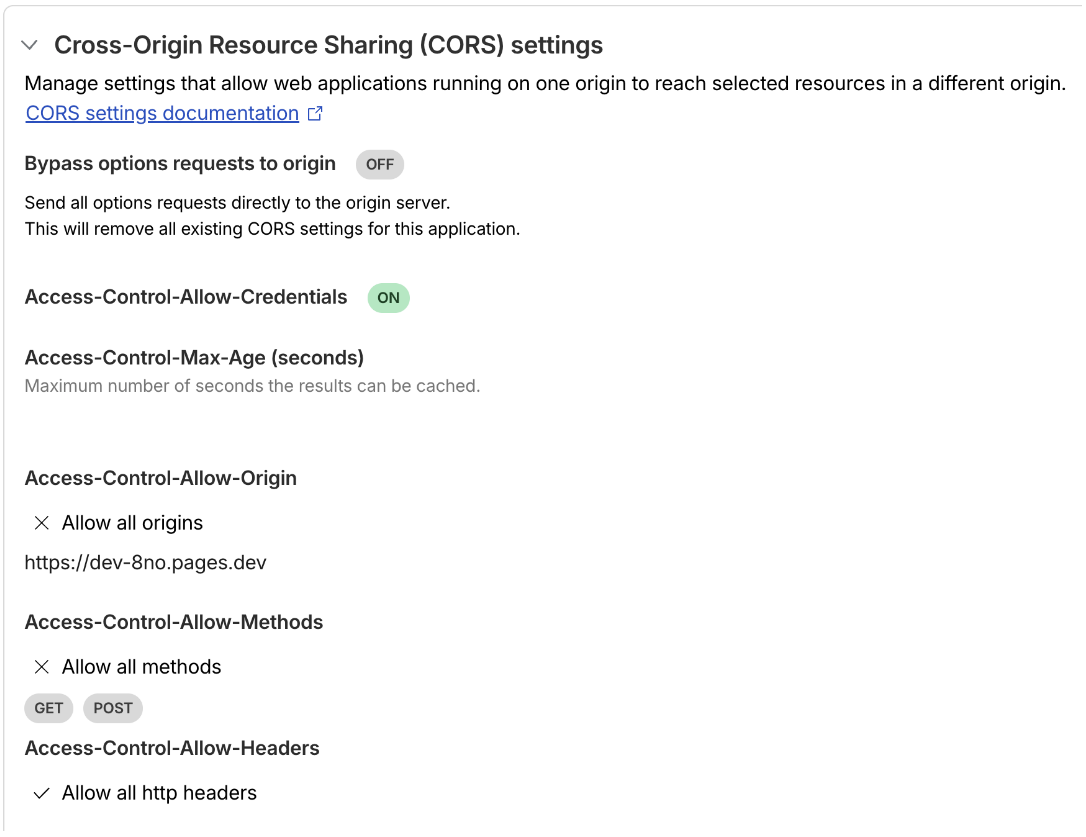
Session Duration (Required): 24 hours

Public hostname

Input method	Domain
Default	deployment-api.manuel-hanifl.workers.dev

Input method	Domain
Default	dev-8no.pages.dev

Figure 10: Cloudflare Zero Trust Access configuration


Cross-Origin Resource Sharing (CORS) settings

Manage settings that allow web applications running on one origin to reach selected resources in a different origin.
CORS settings documentation

Bypass options requests to origin OFF

Send all options requests directly to the origin server.
This will remove all existing CORS settings for this application.

Access-Control-Allow-Credentials ON

Access-Control-Max-Age (seconds)
Maximum number of seconds the results can be cached.

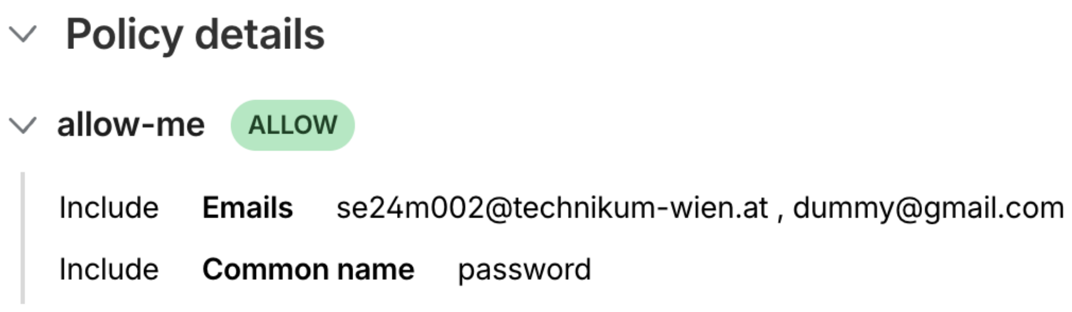
Access-Control-Allow-Origin
✗ Allow all origins
https://dev-8no.pages.dev

Access-Control-Allow-Methods
✗ Allow all methods
GET POST

Access-Control-Allow-Headers
✓ Allow all http headers

Figure 11: Cloudflare Zero Trust Access CORS configuration

CORS is restricted to the deployment dashboard's origin to prevent other origins from accessing the API. It also restricts to only allow the required methods (GET, POST) as no other methods are needed.


Policy details

allow-me ALLOW

Include Emails se24m002@technikum-wien.at , dummy@gmail.com

Include Common name password

Figure 12: Cloudflare Policy configuration

Allowed emails are restricted to only one dummy email and the fh email as no these are the only ones used for testing.

4.4 Used Services

- **Cloudflare Pages:** is used to host both the deployment dashboard and the deployed applications. Pages provides amanged build and deployment pipeline that builds static assets from a connected GitHub repository and deploys them automatically. In this project, deployments are triggered via Pages deploy hooks and are based on dedicated release branches.

- **Cloudflare Workers:** is used to implement the backend Deployment API. The Worker exposes REST endpoints for retrieving commits, triggering deployments, qerying deployment status and accessing deployment history. It orchestrates the deployment process by interacting with the GitHub API and Cloudflare Pages deploy hooks. The worker also validates authentication tokens on each request.
- **Cloudflare Zero Trust Access:** is used to protect the backend from unauthenticated access. Requests to the Worker are only allowed after successful user authentication via One-Time Password (OTP) login.
- **Cloudflare KV:** is used as a serverless key-value store to persist deployment logs. Each deployment action is logged with a timestamp, user information, target app, commit hash, environment and status. The stored data is used to provide deployment history in the dashboard.

5 Security Considerations

The Deployment API exposes functionality to trigger application deployments and therefore must not be publicly accessible.

Authentication is enforced using Cloudflare Zero Trust Access. All requests to the Deployment API require a successful one-time password (OTP) login. Unauthenticated requests are blocked before reaching the backend.

In addition, the Deployment API enforces authentication itself. For every non-preflight request, the backend validates the Cloudflare-issued JSON Web Token (JWT), including signature verification (JWKS), expiration checks, and audience validation (AUD). Requests without a valid token are rejected by the backend.

```
let claims = null;
try {
  claims = await requireAuth(request);
} catch (e) {
  return json(401, { error: 'unauthorized', reason: String(e.message) })
      ;
}
```

Listing 1: Backend enforcement of authentication for all requests

The frontend does not contain any credentials. All deployment logic and security validation are executed exclusively in the backend API.

The Deployment API uses scoped API tokens with minimal permissions. Tokens are limited to the required operations (GitHub repository reference updates, Cloudflare Pages deployment metadata access, and deploy hook triggering). Secrets such as API tokens and deploy hook URLs are stored as Worker secrets and are not exposed to the client.

All communication between the dashboard and the backend API is performed over HTTPS. Cross-origin access is restricted to the Deployment Dashboard origin. Credentialed cross-origin requests are only accepted from explicitly allowed origins.

All user-provided input is validated before being processed by the backend.

```
if (!ALLOWED_ENVS.includes(envName)) {
  return json(400, { error: "Invalid environment" });
}
if (!shaRe.test(commitIn)) {
  return json(400, { error: "Invalid commit SHA" });
}
```

Listing 2: Validation of deployment requests in the Deployment API

More advanced security features such as fine-grained role-based access control, approval workflows, and delegated authorization are intentionally not implemented in this proof of concept.

References

[1] *Limits · Cloudflare Pages docs.* Accessed 2026-01-14. URL: `https://developers.cloudflare.com/pages/platform/limits/`.

[2] *Pricing · Cloudflare Workers docs.* Accessed 2026-01-14. URL: `https://developers.cloudflare.com/workers/platform/pricing/`.

[3] *Workers & Pages Pricing | Cloudflare.* Accessed 2026-01-14. URL: `https://www.cloudflare.com/de-de/plans/zero-trust-services/`.

[4] *Workers KV Pricing | Cloudflare Developers.* Accessed 2026-01-14. URL: `https://developers.cloudflare.com/kv/platform/pricing/`.

List of Figures