Cloud-Based Deployment Platform
Proof of Concept Documentation

Manuel Hanifl
MSE-BB-3-WS2025-CPS

January 23, 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

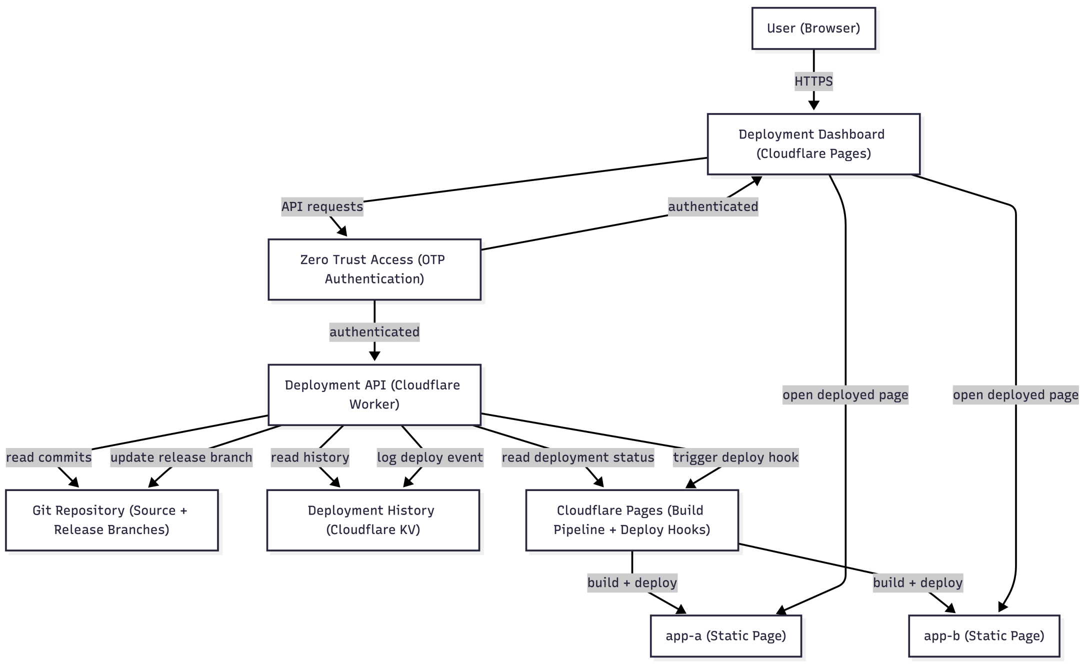


Figure 1: Architecture Diagram

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.

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

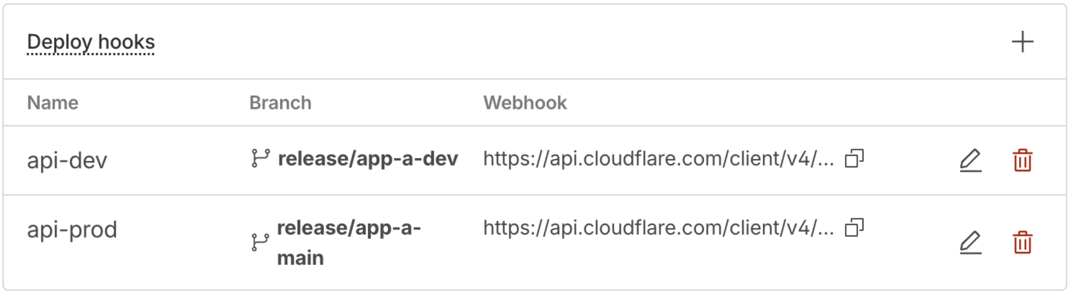
This proof of concept was implemented entirely within the free tiers of the used cloud services.

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].

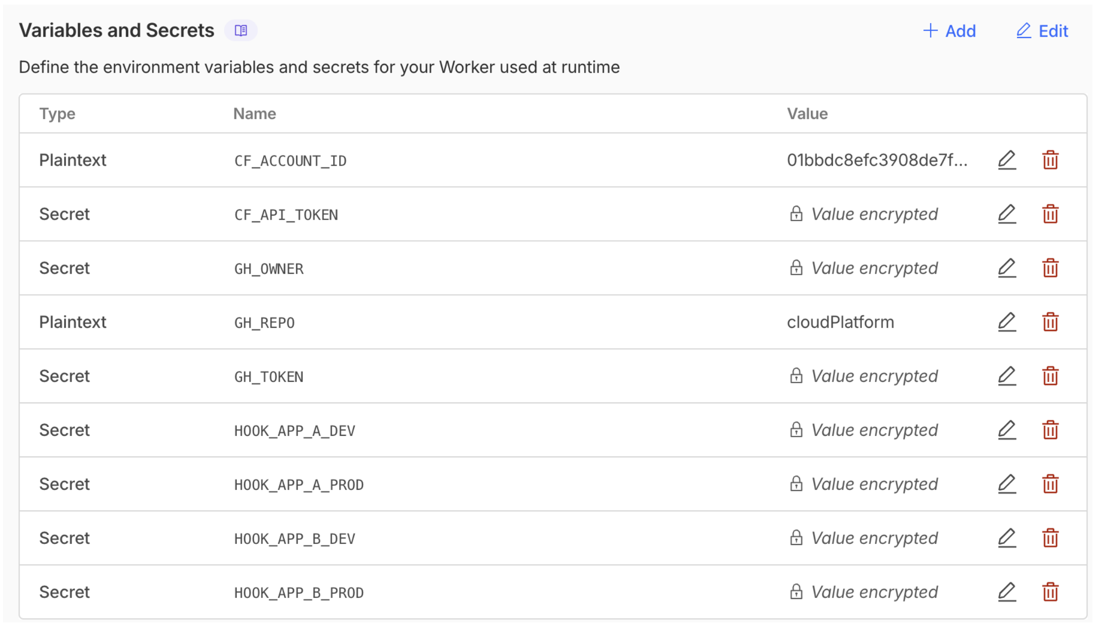
4.3 Configuration of Cloud Services



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 2: Cloudflare Pages deploy hook configuration

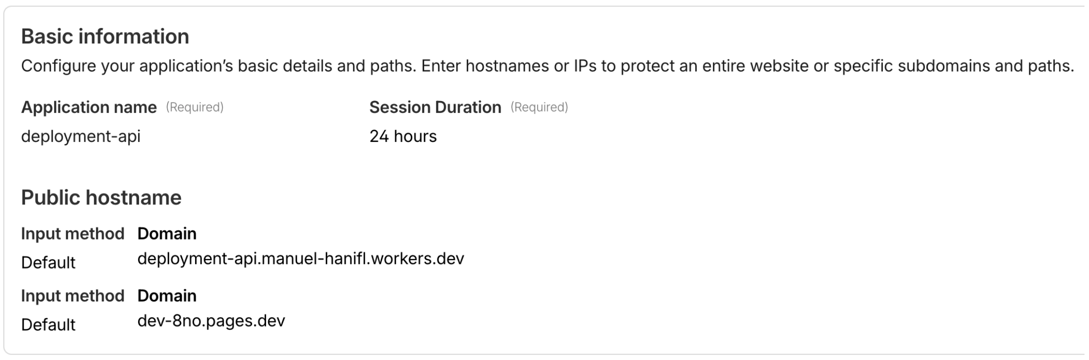


Variables and Secrets

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 3: Cloudflare Worker API variables



Basic information

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

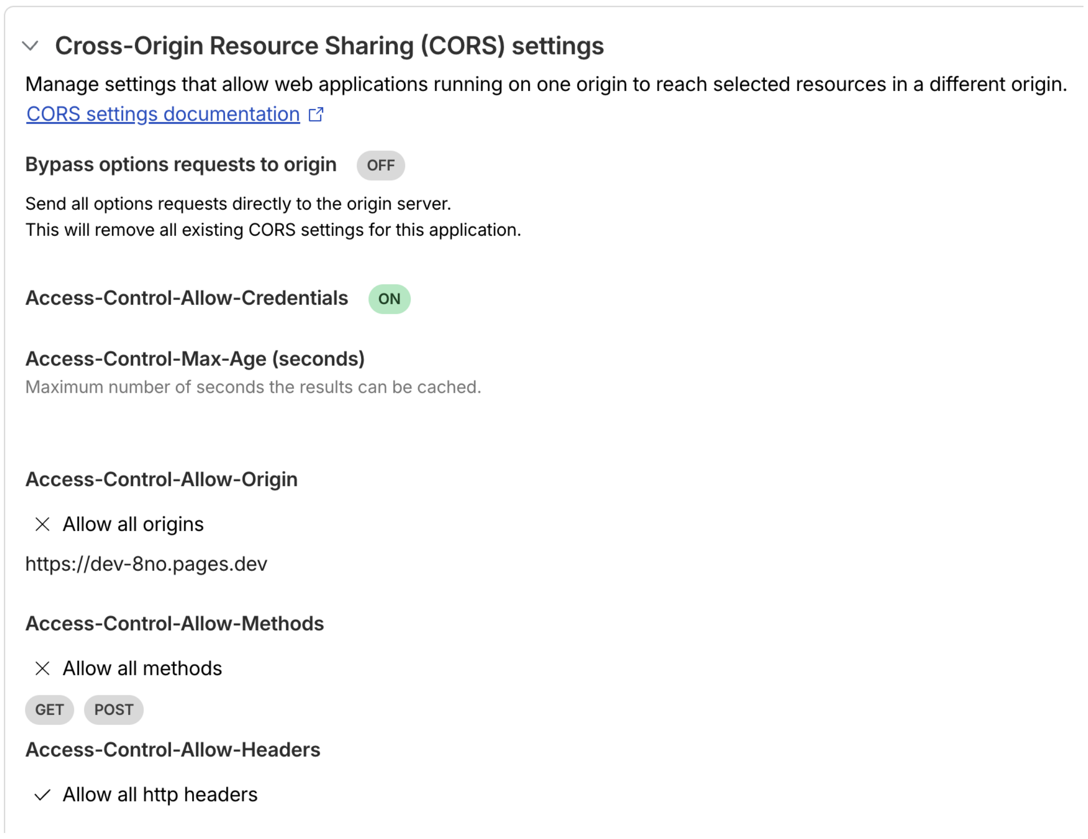
Application name (Required)	Session Duration (Required)
deployment-api	24 hours

Public hostname

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

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

Figure 4: 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

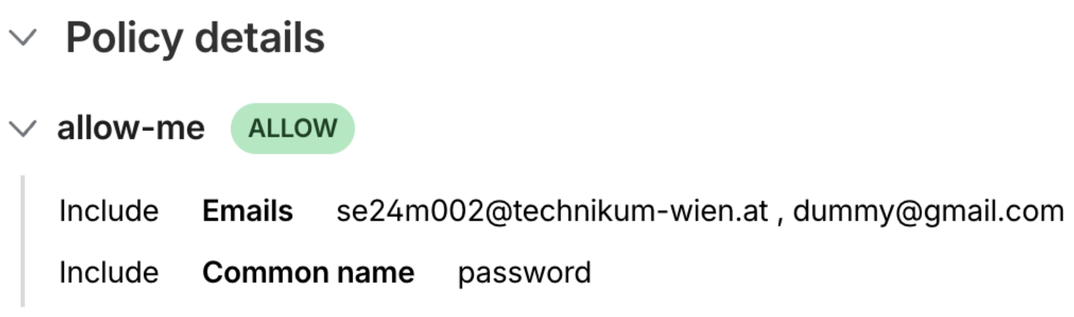
✗ Allow all methods

GET POST

Access-Control-Allow-Headers

✓ Allow all http headers

Figure 5: Cloudflare Zero Trust Access CORS configuration



Policy details

allow-me ALLOW

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

Include Common name password

Figure 6: Cloudflare Policy configuration

4.4 Used Services

- **Cloudflare Pages:** Used for static hosting of the Deployment Dashboard and the deployable applications.
- **Cloudflare Workers:** Used to implement the Deployment API that validates requests, triggers deployments, and retrieves deployment metadata.
- **Cloudflare Zero Trust Access:** Used to authenticate users and protect the deployment API from unauthenticated access.
- **Cloudflare KV:** Used to persist deployment history events.

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.

The frontend does not contain any credentials. All deployment logic is executed exclusively in the backend API.

The Deployment API uses scoped API tokens with minimal permissions. Tokens are scoped to only the required APIs (GitHub repo ref update; Cloudflare Pages deployment read; Pages deploy hook trigger). 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. All user provided content is validated before being processed.

```
      if (!ALLOWED_ENVS.includes(envName)) {
        return json(400, { error: ‘Invalid env.‘ });
      }
      if (!shaRe.test(commitIn)) {
        return json(400, { error: "Invalid commit SHA. Expected 7 40
           hex characters." });
        }
```
Listing 1: Validation of deployment requests in the Deployment API

More advanced security features such as role-based access control, approval workflows, and audit logging are 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