REST API

- How to use the API
  - Get an API token
  - Make requests
- Examples
  - Triggering a new build every hour
  - Deleting old deployments after a week
  - Sharing project information
- Related resources

import { APIRequest, DashButton } from "~/components";

The Pages API empowers you to build automations and integrate Pages with your development workflow. At a high level, the API endpoints let you manage deployments and builds and configure projects. Cloudflare supports Deploy Hooks for headless CMS deployments. Refer to the API documentation for a full breakdown of object types and endpoints.

How to use the API

Get an API token

To create an API token:

1. In the Cloudflare dashboard, go to the **Account API tokens** page.
2. Select **Create Token**.
3. You can go to **Edit Cloudflare Workers** template > **Use template** or go to **Create Custom Token** > **Get started**. If you create a custom token, you will need to make sure to add the **Cloudflare Pages** permission with **Edit** access.

Make requests

After creating your token, you can authenticate and make requests to the API using your API token in the request headers. For example, here is an API request to get all deployments in a project.

Try it with one of your projects by replacing `{account_id}`, `{project_name}`, and `<API_TOKEN>`. Refer to Find your account ID for more information.

Examples

The API is even more powerful when combined with Cloudflare Workers: the easiest way to deploy serverless functions on Cloudflare's global network. The following section includes three code examples on how to use the Pages API. To build and deploy these samples, refer to the Get started guide.

Triggering a new build every hour

Suppose we have a CMS that pulls data from live sources to compile a static output. You can keep the static content as recent as possible by triggering new builds periodically using the API.

```
const endpoint =
"https://api.cloudflare.com/client/v4/accounts/{account_id}/pages/projects/{project_name}/deployments";

export default {
async scheduled(_, env) {
const init = {
method: "POST",
headers: {
"Content-Type": "application/json;charset=UTF-8",
// We recommend you store the API token as a secret using the Workers dashboard or using Wrangler as documented here:
https://developers.cloudflare.com/workers/wrangler/commands/general/#secret
Authorization: `Bearer ${env.API_TOKEN}`,
},
};

await fetch(endpoint, init);
},
```

```
};
```

After you have deployed the JavaScript Worker, set a cron trigger in your Worker to run this script periodically. Refer to Cron Triggers for more details.

Deleting old deployments after a week

Cloudflare Pages hosts and serves all project deployments on preview links. Suppose you want to keep your project private and prevent access to your old deployments. You can use the API to delete deployments after a month, so that they are no longer public online. The latest deployment for a branch cannot be deleted.

:::note To delete individual deployments from the command line, you can use `wrangler pages deployment delete`. :::

```
const endpoint =
"https://api.cloudflare.com/client/v4/accounts/{account_id}/pages/projects/{project_name}/deployments";
const expirationDays = 7;

export default {
async scheduled(_, env) {
const init = {
headers: {
"Content-Type": "application/json;charset=UTF-8",
// We recommend you store the API token as a secret using the Workers dashboard or using Wrangler as documented here:
https://developers.cloudflare.com/workers/wrangler/commands/general/#secret
Authorization: `Bearer ${env.API_TOKEN}`,
},
};

const response = await fetch(endpoint, init);
const deployments = await response.json();

for (const deployment of deployments.result) {
// Check if the deployment was created within the last x days (as defined by `expirationDays` above)
if (
(Date.now() - new Date(deployment.created_on)) / 86400000 >
```

```
    expirationDays
  ) {
    // Delete the deployment
    await fetch(`${endpoint}/${deployment.id}`, {
      method: "DELETE",
      headers: {
        "Content-Type": "application/json;charset=UTF-8",
        Authorization: `Bearer ${env.API_TOKEN}`,
      },
    });
  }
 }
},
};
```

After you have deployed the JavaScript Worker, you can set a cron trigger in your Worker to run this script periodically. Refer to the Cron Triggers guide for more details.

Sharing project information

Imagine you are working on a development team using Pages to build your websites. You would want an easy way to share deployment preview links and build status without having to share Cloudflare accounts. Using the API, you can easily share project information, including deployment status and preview links, and serve this content as HTML from a Cloudflare Worker.

```
const deploymentsEndpoint =
 "https://api.cloudflare.com/client/v4/accounts/{account_id}/pages/projects/{project_name}/deployments";
const projectEndpoint =
 "https://api.cloudflare.com/client/v4/accounts/{account_id}/pages/projects/{project_name}";

export default {
 async fetch(request, env) {
  const init = {
    headers: {
      "content-type": "application/json;charset=UTF-8",
      // We recommend you store the API token as a secret using the Workers dashboard or using Wrangler as documented here:
      https://developers.cloudflare.com/workers/wrangler/commands/general/#secret
```

```
        Authorization: `Bearer ${env.API_TOKEN}`,
    },
};

const style = `body { padding: 6em; font-family: sans-serif; } h1 { color: #f6821f }`;
let content = "<h2>Project</h2>";

let response = await fetch(projectEndpoint, init);
const projectResponse = await response.json();
content += `<p>Project Name: ${projectResponse.result.name}</p>`;
content += `<p>Project ID: ${projectResponse.result.id}</p>`;
content += `<p>Pages Subdomain: ${projectResponse.result.subdomain}</p>`;
content += `<p>Domains: ${projectResponse.result.domains}</p>`;
content += `<a href="${projectResponse.result.canonical_deployment.url}"><p>Latest preview:
${projectResponse.result.canonical_deployment.url}</p></a>`;

content += `<h2>Deployments</h2>`;
response = await fetch(deploymentsEndpoint, init);
const deploymentsResponse = await response.json();

for (const deployment of deploymentsResponse.result) {
    content += `<a href="${deployment.url}"><p>Deployment: ${deployment.id}</p></a>`;
}

let html = `
      <!DOCTYPE html>
      <head>
        <title>Example Pages Project</title>
      </head>
      <body>
        <style>${style}</style>
        <div id="container">
          ${content}
        </div>
      </body>`;

return new Response(html, {
    headers: {
        "Content-Type": "text/html;charset=UTF-8",
```

```
□□□},
□□});
□},
};
```

Related resources

- Pages API Docs
- Workers Getting Started Guide
- Workers Cron Triggers

□□□ #2
□ □□□ □□□ 2026-11-01 02:59:22 UTC
□ □□□ □□□ 2026-11-01 02:59:24 UTC