SVELTEKIT • BUILD AND DEPLOY

Cloudflare Pages

ON THIS PAGE

To deploy to Cloudflare Pages, use `adapter-cloudflare`.

This adapter will be installed by default when you use `adapter-auto`. If you plan on staying with Cloudflare Pages, you can switch from `adapter-auto` to using this adapter directly so that values specific to Cloudflare Workers are emulated during local development, type declarations are automatically applied, and the ability to set Cloudflare-specific options is provided.

Comparisons

- `adapter-cloudflare` – supports all SvelteKit features; builds for Cloudflare Pages
- `adapter-cloudflare-workers` – supports all SvelteKit features; builds for Cloudflare Workers
- `adapter-static` – only produces client-side static assets; compatible with Cloudflare Pages

Usage

Install with `npm i -D @sveltejs/adapter-cloudflare`, then add the adapter to your `svelte.config.js`:

svelte.config.js

```
    adapter: adapter({
      // See below for an explanation of these options
      routes: {
        include: ['/*'],
        exclude: ['<all>']
      },
      platformProxy: {
        configPath: 'wrangler.toml',
        environment: undefined,
        experimentalJsonConfig: false,
        persist: false
      }
    })
  }
};
```

Options

routes

Allows you to customise the `_routes.json` file generated by `adapter-cloudflare`.

`include` defines routes that will invoke a function, and defaults to `['/*']`

`exclude` defines routes that will *not* invoke a function — this is a faster and cheaper way to serve your app's static assets. This array can include the following special values:

`<build>` contains your app's build artifacts (the files generated by Vite)

`<files>` contains the contents of your `static` directory

`<prerendered>` contains a list of prerendered pages

`<all>` (the default) contains all of the above

You can have up to 100 `include` and `exclude` rules combined. Generally you can omit the `routes` options, but if (for example) your `<prerendered>` paths exceed that limit, you may find it helpful to manually create an `exclude` list that includes `'/articles/*'` instead of

Preferences for the emulated `platform.env` local bindings. See the getPlatformProxy Wrangler API documentation for a full list of options.

Deployment

Please follow the Get Started Guide for Cloudflare Pages to begin.

When configuring your project settings, you must use the following settings:

Framework preset – SvelteKit

Build command – `npm run build` or `vite build`

Build output directory – `.svelte-kit/cloudflare`

Runtime APIs

The `env` object contains your project's bindings, which consist of KV/DO namespaces, etc. It is passed to SvelteKit via the `platform` property, along with `context`, `caches`, and `cf`, meaning that you can access it in hooks and endpoints:

```
export async function POST({ request, platform }) {

  const x = platform.env.YOUR_DURABLE_OBJECT_NAMESPACE.idFromName('x');
}
```

SvelteKit's built-in `$env` module should be preferred for environment variables.

To include type declarations for your bindings, reference them in your `src/app.d.ts`:

src/app.d

```
declare global {
  namespace App {
```

```
      };
    }
  }
}

export {};
```

Testing Locally

Cloudflare Workers specific values in the `platform` property are emulated during dev and preview modes. Local bindings are created based on the configuration in your `wrangler.toml` file and are used to populate `platform.env` during development and preview. Use the adapter config `platformProxy` option to change your preferences for the bindings.

For testing the build, you should use wrangler version 3. Once you have built your site, run `wrangler pages dev .svelte-kit/cloudflare`.

Notes

Functions contained in the `/functions` directory at the project's root will *not* be included in the deployment, which is compiled to a single `_worker.js` file. Functions should be implemented as server endpoints in your SvelteKit app.

The `_headers` and `_redirects` files specific to Cloudflare Pages can be used for static asset responses (like images) by putting them into the `/static` folder.

However, they will have no effect on responses dynamically rendered by SvelteKit, which should return custom headers or redirect responses from server endpoints or with the `handle` hook.

Troubleshooting

You may wish to refer to Cloudflare's documentation for deploying a SvelteKit site.

Accessing the file system

You can't use `fs` in Cloudflare Workers — you must prerender the routes in question.

Edit this page on GitHub

PREVIOUS
Single-page apps

NEXT
Cloudflare Workers