SVELTEKIT • BUILD AND DEPLOY

Cloudflare Workers

ON THIS PAGE

To deploy to Cloudflare Workers, use `adapter-cloudflare-workers`.

> Unless you have a specific reason to use `adapter-cloudflare-workers`, it's recommended that you use `adapter-cloudflare` instead. Both adapters have equivalent functionality, but Cloudflare Pages offers features like GitHub integration with automatic builds and deploys, preview deployments, instant rollback and so on.

Usage

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

svelte.config.js

```
import adapter from '@sveltejs/adapter-cloudflare-workers';

export default {
  kit: {
    adapter: adapter({
      config: 'wrangler.toml',
      platformProxy: {
        configPath: 'wrangler.toml',
        environment: undefined,
        experimentalJsonConfig: false,
        persist: false
      }
    })
  }
```

config

Path to your custom `wrangler.toml` config file.

platformProxy

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

Basic Configuration

This adapter expects to find a wrangler.toml file in the project root. It should look something like this:

wrangler.toml

```
name = "<your-service-name>"
account_id = "<your-account-id>"

main = "./.cloudflare/worker.js"
site.bucket = "./.cloudflare/public"

build.command = "npm run build"

compatibility_date = "2021-11-12"
workers_dev = true
```

`<your-service-name>` can be anything. `<your-account-id>` can be found by logging into your Cloudflare dashboard and grabbing it from the end of the URL:

```
https://dash.cloudflare.com/<your-account-id>
```

You should add the `.cloudflare` directory (or whichever directories you specified for `main`

```
npm i -g wrangler
wrangler login
```

Then, you can build your app and deploy it:

```
wrangler deploy
```

Custom config

If you would like to use a config file other than `wrangler.toml` you can specify so using the `config` option.

If you would like to enable Node.js compatibility, you can add "nodejs_compat" flag to `wrangler.toml`:

wrangler.toml

```
compatibility_flags = [ "nodejs_compat" ]
```

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');
}
```

src/app.d

```
declare global {
  namespace App {
    interface Platform {
      env?: {
        YOUR_KV_NAMESPACE: KVNamespace;
        YOUR_DURABLE_OBJECT_NAMESPACE: DurableObjectNamespace;
      };
    }
  }
}

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 dev .`

Troubleshooting

Worker size limits

When deploying to workers, the server generated by SvelteKit is bundled into a single file. Wrangler will fail to publish your worker if it exceeds the size limits after minification. You're unlikely to hit this limit usually, but some large libraries can cause this to happen. In that case, you can try to reduce the size of your worker by only importing such libraries on

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

Edit this page on GitHub

PREVIOUS
Cloudflare Pages

NEXT
Netlify