Mux Video

This example uses Mux Video, an API-first platform for video. The example features video uploading and playback in a Next.js application.

Demo

https://with-mux-video.vercel.app/

This project was used to create stream.new

Deploy your own

Deploy the example using Vercel:

Deploy

How to use

Execute `create-next-app` with npm or Yarn to bootstrap the example:

```
npx create-next-app --example with-mux-video with-mux-video-app
# or
yarn create next-app --example with-mux-video with-mux-video-app
# or
pnpm create next-app -- --example with-mux-video with-mux-video-app
```

Note

Important: When creating uploads, this demo sets `cors_origin: "*"` in the `pages/api/upload.js` file. For extra security, you should update this value to be something like `cors_origin: 'https://your-app.com'`, to restrict uploads to only be allowed from your application.

This example uses:

- SWR — dynamically changing the `refreshInterval` depending on if the client should be polling for updates or not
- `/pages/api` routes — a couple endpoints for making authenticated requests to the Mux API.
- Dynamic routes using `getStaticPaths` and `fallback: true`, as well as dynamic API routes.

Configuration

Step 1. Create an account in Mux

All you need to set this up is a Mux account. You can sign up for free and pricing is pay-as-you-go. There are no upfront charges, you get billed monthly only for what you use.

Without entering a credit card on your Mux account all videos are in "test mode" which means they are watermarked and clipped to 10 seconds. If you enter a credit card all limitations are lifted and you get $20 of free credit. The free credit should be plenty for you to test out and play around with everything before you are charged.

Step 2. Set up environment variables

Copy the `.env.local.example` file in this directory to `.env.local` (which will be ignored by Git):

```
cp .env.local.example .env.local
```

Then, go to the settings page in your Mux dashboard set each variable on `.env.local`, get a new **API Access Token** and set each variable in `.env.local`:

- `MUX_TOKEN_ID` should be the `TOKEN ID` of your new token
- `MUX_TOKEN_SECRET` should be `TOKEN SECRET`

Step 3. Deploy on Vercel

You can deploy this app to the cloud with Vercel (Documentation).

To deploy on Vercel, you need to set the environment variables using Vercel CLI (Documentation).

Install the Vercel CLI, log in to your account from the CLI, and run the following commands to add the environment variables. Replace the values with the corresponding strings in `.env.local`:

```
vercel secrets add next_example_mux_token_id <MUX_TOKEN_ID>
vercel secrets add next_example_mux_token_secret <MUX_TOKEN_SECRET>
```

Then push the project to GitHub/GitLab/Bitbucket and import to Vercel to deploy.