**Fleek CLI**

## Deploy Your Static Web App Using the Fleek CLI

​

This guide walks you through deploying a static web application, such as a Next.js project, to the Fleek platform using the Fleek Command Line Interface (CLI). Utilizing the CLI provides greater transparency and control over the build and deployment process compared to the web UI, allowing you to verify the exact code bundle that gets deployed.

​

### Installing the Fleek CLI

​

First, you need to install the Fleek CLI globally on your system using npm. Open your terminal and run the following command:

​

```bash

npm install -g @fleek-platform/cli

The Fleek CLI is a powerful tool enabling interaction with Fleek's hosting, storage, and other services directly from your command line. Installing it globally (-g) ensures the fleek command is accessible from any directory in your terminal.

**Why Use the Fleek CLI?**

Deploying directly from your terminal using the Fleek CLI offers a distinct advantage: it allows you to test the precise build process locally that Fleek will use. This gives you confidence that what you test locally is exactly what gets deployed, offering more insight than the potentially opaque build environment within the web UI.

**Pre-Deployment Preparations**

Before initiating the deployment, perform these essential checks and setup steps within your project directory:

1. **Install Dependencies:** Ensure all project dependencies are correctly installed. Fleek deployments often work more reliably when using npm. Run:

npm install

This helps replicate the environment Fleek might use, preventing failures caused by missing packages.

1. **Verify Local Build:** Confirm that your application builds successfully locally. For a Next.js project, this typically involves:

npm run build

This command usually executes next build internally. A successful local build is crucial to catch potential errors before attempting deployment.

1. **Configure TypeScript (If Applicable):** To prevent test files or other non-essential files from being included in your production build, modify your tsconfig.json. Add the relevant patterns to the exclude array. For example, to exclude a test directory:

// tsconfig.json

{

// ... other configurations

"exclude": [

"node\_modules",

"test/\*\*/\*" // Ensure test files are excluded

]

// ... other configurations

}

The tsconfig.json file directs the TypeScript compiler. The exclude array specifies files and directories to ignore during compilation, leading to cleaner, smaller production builds.

1. **Confirm Static Output:** Check the output of your build command (npm run build). For static Next.js sites compatible with Fleek's basic hosting, look for confirmation messages indicating static pre-rendering, like (Static) prerendered as static content. Fleek excels at hosting static sites composed of HTML, CSS, and JavaScript files generated during the build process. Ensure your build process outputs to a specific directory (commonly ./out for static Next.js exports).

**Deploying with the Fleek CLI**

With preparations complete, you can now use the Fleek CLI to deploy your site:

1. **Login to Fleek:** Authenticate the CLI with your Fleek account.

fleek login

This command will output a unique URL. Open this URL in your browser, log in to your Fleek account if necessary, and authorize the CLI connection. Your terminal will display a confirmation message upon successful authentication.

1. **Initialize Site Configuration:** Navigate to your project's root directory in the terminal and initialize the Fleek site configuration:

fleek sites init

The CLI will guide you through the following prompts:

* + **Select Project:** Choose the Fleek project you want to deploy to (e.g., "First Project").
  + **Link Existing Site?:** Choose "no" to create a new site deployment configuration.
  + **Site Name:** Enter a name for your new site (e.g., "tsender-2"). This name is used within the Fleek dashboard.
  + **Dist Directory:** Specify the directory containing your built static site files. For static Next.js exports, this is typically ./out.
  + **Include Build Command?:** Choose "yes".
  + **Build Command:** Enter the command required to build your site. The default npm run build is often suitable.
  + **Configuration Format:** Select JSON (fleek.config.json). This format is generally reliable.

This process creates a fleek.config.json file in your project's root directory.

1. **Review**fleek.config.json**:** Inspect the generated configuration file. It defines how the Fleek CLI should handle your deployment. It will look similar to this:

{

"sites": [

{

"slug": "some-unique-autogenerated-slug", // Fleek auto-generates this for the URL

"distDir": "./out", // Directory containing build output

"buildCommand": "npm run build" // Command to build the site

}

]

}

This file tells the CLI which command to run for building (buildCommand) and which directory's contents to upload (distDir).

1. **Deploy the Site:** Execute the deployment command:

fleek sites deploy

The CLI will perform the following actions:

* + Run the buildCommand specified in fleek.config.json (e.g., npm run build) locally.
  + Upload the contents of the specified distDir (e.g., ./out) to IPFS via Fleek.
  + Assign a unique Fleek URL based on the generated slug.

Deployment can take a few minutes depending on the size of your site and network conditions. Upon successful completion, the CLI will output:

* + A success message (e.g., "Success! Deployed!").
  + The Site IPFS Content Identifier (CID).
  + The gateway URL for accessing your deployed site (e.g., https://some-unique-autogenerated-slug.on-fleek.app).

**Verifying Your Deployment**

After deployment, verify that your site is live and functioning correctly:

1. **Access the Site:** Open the gateway URL provided by the CLI in your web browser. Your application should load.
2. **Test Functionality:** If your application includes web3 interactions (like connecting a wallet or interacting with smart contracts):
   * Connect your web3 wallet (e.g., MetaMask).
   * Perform key actions within your application (e.g., submitting a form, triggering a transaction).
   * **Troubleshooting:** Be aware that browser security features or extensions (like Brave Shields) might block requests from your deployed HTTPS site to local RPC endpoints (e.g., http://127.0.0.1:8545 for a local testnet like Anvil or Hardhat). If you encounter connection errors in the browser console during testing with local nodes, try disabling shields or security features specifically for your deployed site's URL.
   * Confirm that transactions are processed as expected by checking wallet prompts and blockchain state (if applicable).

Successful interaction confirms that your statically deployed site retains its dynamic web3 capabilities.

**Fleek UI Confirmation and Next Steps**

You can also view and manage your deployment through the Fleek web dashboard:

1. **Check Dashboard:** Navigate to your project within the Fleek dashboard (fleek.xyz). You should see your newly deployed site listed (e.g., "tsender-2").
2. **View Details:** Clicking on the site reveals deployment history (showing it was deployed via CLI), build logs, the associated IPFS CID, and site settings.
3. **Custom Domains:** To use a custom domain instead of the default .on-fleek.app subdomain, navigate to the site's Settings -> Domains section in the Fleek UI and follow the instructions to add and configure your domain (e.g., your-app.com).

**Conclusion**

Congratulations! You have successfully deployed a static web application to Fleek using the Fleek CLI. This method provides excellent control over the build process and ensures your application is deployed reliably to decentralized storage on IPFS, ready to be accessed globally. You can now further enhance your application or configure a custom domain via the Fleek dashboard.

​