**Fleek UI  
Deploying Your Next.js App to Fleek via the UI**

After building and testing your application, the final crucial step is deployment. Making your application live allows judges (in a hackathon scenario) or users to interact with it. While numerous deployment platforms exist, this lesson focuses on deploying a Next.js application as a static site using the Fleek UI.

**Introducing Fleek: Your web3 Hosting Platform**

Fleek (fleek.xyz) is a platform specializing in building and hosting static websites, making it an excellent choice for web3 applications. Key reasons for using Fleek include:

* **Static Site Focus:** Optimized for deploying front-ends built as static sites.
* **web3 Integrations:** Offers various plugins and integrations tailored for the web3 ecosystem.
* **Fleek Functions:** Provides serverless function capabilities for adding dynamic elements later (though not covered in this specific deployment).

These features make Fleek a preferred choice for hosting decentralized application front-ends.

**Static vs. Dynamic Sites: Why Go Static for web3?**

Understanding the difference between static and dynamic sites is vital for web3 deployment:

* **Static Site:** Consists of pre-built HTML, CSS, and JavaScript files served directly to the user's browser. All logic runs client-side. There's no server-side computation required after the initial build. This aligns well with web3 principles like decentralization (the front-end is often open-source, computation happens on the user's machine), reduces infrastructure costs for developers, and simplifies hosting on platforms like IPFS.
* **Dynamic Site:** Involves a server that processes requests, potentially interacts with databases, and renders content on the backend before sending it to the user. This adds complexity and ongoing server costs.

For optimal compatibility with Fleek's static hosting and decentralized storage like IPFS, we need to configure our Next.js application to build as a 100% static site.

**Configuring Next.js for Static Export**

To ensure our Next.js application builds as a static site, we need to modify the next.config.ts file in the root of our project. Add or update the configuration to include the following options:

import type { NextConfig } from "next";

​

const nextConfig: NextConfig = {

// Instructs Next.js to output static HTML/CSS/JS files

output: 'export',

​

// Specifies the directory for the static output (Fleek often expects 'out')

distDir: 'out',

​

// Required for static export as Next.js image optimization needs a server

images: {

unoptimized: true,

},

​

// Set the base path if deploying to a subdirectory (empty for root)

basePath: "",

​

// Ensures assets are referenced correctly in static exports, especially for IPFS

assetPrefix: "./",

​

// Adds trailing slashes to URLs (e.g., /about/), often expected by static hosts like Fleek

trailingSlash: true

};

​

export default nextConfig;

These settings ensure that running the build command generates a self-contained static application suitable for deployment on platforms like Fleek.

**Verifying Your Static Build Locally**

Before attempting to deploy, it's crucial to verify that your application builds correctly as a static site with the new configuration. Run the build command in your project's terminal:

# If using pnpm

pnpm run build

​

# If using npm

npm run build

​

# If using yarn

yarn build

Watch the terminal output. A successful build will complete without errors and often indicates which pages were pre-rendered as static content. This local check helps catch configuration errors early, saving time during deployment.

**Preparing Your Code on GitHub**

Fleek pulls code directly from a Git provider, with GitHub being the primary supported option. Using version control like Git and hosting your code on GitHub is standard practice and highly recommended.

1. **Create a GitHub Repository:** Go to GitHub and create a new **public** repository for your project (e.g., my-web3-app-ui).
2. **Initialize Git Locally (if needed):** If your project isn't already a Git repository, open your terminal in the project root and run git init.
3. **Check Status and**.gitignore**:** Run git status. Ensure that sensitive files (like .env.local), node\_modules, and build output directories (.next, out) are listed in your .gitignore file and are not being tracked.
4. **Stage and Commit:**

git add .

git commit -m "Configure for static export and prepare for Fleek deployment"

1. **Link to Remote Repository:**

# Replace <GITHUB\_REPOSITORY\_URL> with the URL of the repo you created

git remote add origin <GITHUB\_REPOSITORY\_URL>

1. **Push to GitHub:**

# Pushes the main branch to your remote GitHub repository

git push --set-upstream origin main

*(Note: You might need to authenticate with GitHub via the command line, a Personal Access Token (PAT), or SSH keys if you haven't done so previously).*

Your code, including the next.config.ts changes, is now on GitHub, ready for Fleek.

**Setting Up Fleek and Connecting GitHub**

Now, let's connect Fleek to your GitHub repository:

1. **Log in to Fleek:** Go to fleek.xyz and log in or sign up.
2. **Navigate to Hosting:** Go to your Dashboard and select the **Hosting** tab.
3. **Start Deployment:** Click **"Add new"** and then **"Deploy my site"**.
4. **Connect Git Provider:** Select **GitHub**.
5. **Authorize Fleek:** You'll be redirected to GitHub to authorize the "Fleek CI" application. Review the permissions (initially, it might just request email access) and authorize it.
6. **Install Fleek GitHub App:** You'll then be prompted to install the Fleek GitHub App on your account or organization.
7. **Select Repositories (Important Security Tip):** Choose **"Only select repositories"** and select the specific repository you just pushed your code to. Avoid granting access to all repositories unless necessary.
8. **Install & Authorize:** Click the "Install & Authorize" button.

You will be redirected back to Fleek, now successfully connected to your GitHub account and selected repository.

**Deploying Your Site via the Fleek UI**

With Fleek connected to your GitHub repository, you can configure the deployment:

1. **Select Repository:** Choose the repository you just authorized (e.g., my-web3-app-ui).
2. **Framework Detection:** Fleek should automatically detect your framework as Next.js Static. If not, you may need to select it manually.
3. **Build Settings:**
   * **Branch:** Ensure the correct branch (main or master) is selected.
   * **Publish Directory:** Verify this is set to out (matching the distDir in your next.config.ts).
   * **Build Command:** Confirm the build command is correct for your package manager (e.g., pnpm install && pnpm run build or npm install && npm run build). Adjust if necessary.
4. **Environment Variables:** This is critical for variables your application needs at runtime (like API keys or project IDs).
   * Find any variables prefixed with NEXT\_PUBLIC\_ in your local .env.local file (e.g., NEXT\_PUBLIC\_WALLETCONNECT\_PROJECT\_ID).
   * In the Fleek UI's "Environment variables" section, enter the variable **name** (e.g., NEXT\_PUBLIC\_WALLETCONNECT\_PROJECT\_ID) and its corresponding **value**.
   * Click **"Add"** for each variable. **Never commit your**.env.local**file to Git.**
5. **Deploy:** Once all settings are confirmed and environment variables are added, click **"Deploy site"**.

**Monitoring Your Fleek Deployment**

Fleek will now begin the deployment process. You'll see status updates like "Queued" and "Building". Fleek provides a temporary domain (e.g., your-site-name.on-fleek.app) where your site will be accessible once deployed.

Navigate to the **Deployments** tab for your site in Fleek to see detailed progress and logs. The typical stages include:

1. **Cloning Git Repository:** Fetching your code from GitHub.
2. **Creating Check Run:** Setting up status checks on GitHub.
3. **Build Logs:** Running the specified install and build commands (pnpm install && pnpm run build). This is where build errors will appear.
4. **Upload to IPFS:** Uploading the contents of your build output directory (out) to the InterPlanetary File System (IPFS), the decentralized storage layer Fleek uses.
5. **Availability Check:** Verifying the uploaded content is accessible.
6. **Release:** Finalizing the deployment and making it live.

Successful completion of these steps means your site is deployed!

**Troubleshooting Common Fleek Deployment Issues**

Sometimes deployments fail. Here's a common issue and how to approach it:

* **Error:** The Fleek UI shows an error like dist directory does not exist or similar regarding the publish directory.
* **Meaning:** This is often a generic UI message indicating that the **build step** (e.g., pnpm run build) failed for *any* reason. It doesn't necessarily mean the directory itself is missing.
* **Solution:** The *real* error is usually hidden within the **Build Logs**. Expand the build log section in the Fleek deployment details page and scroll through it carefully to find the actual error message (e.g., a TypeScript error, a missing dependency, an incorrect environment variable).
* **Alternative (Fleek CLI):** If you consistently face issues with UI deployments or need more detailed error feedback, consider using the **Fleek CLI** (Command Line Interface). Deploying via the CLI often provides more direct and informative error messages, aiding in troubleshooting build problems.