Building NextJS Apps

Two Deployment Options:

- Next.js offers two primary deployment options: Standard Build and Full Static Build.

Standard Build:

- Achieved with next build command.
- Produces an optimised production build with server-side capabilities.
- Requires a Node.js server to run because of server-side features like pre-rendering, API routes, and page revalidation.
- You need to redeploy when your code changes and when content changes that impact pages without revalidation settings.

Full Static Build:

- Achieved with next export command.
- Produces a 100% static application with only HTML, CSS, and JavaScript.
- Does not require a Node.js server for hosting, making it simpler for deployment.
- Not suitable if your site relies on API routes, server-side pages with getServerSideProps, page revalidations, or static paths with fallback set to true or blocking.
- Requires redeployment whenever code or content changes occur since there's no server to handle on-demand tasks.

Considerations:

- Choose the deployment option that aligns with your project's requirements and features.
- Standard Build is suitable for projects with server-side features, while Full Static Build is ideal for purely static sites.
- Be aware of the need for Node.js server hosting when using the Standard Build.
- Understand the implications of each option for redeployment, based on code and content changes.

Deployment Flexibility:

- You can choose the deployment option that best suits your project's needs and constraints.
- Some projects may benefit from the simplicity of Full Static Build, while others may require the server-side capabilities of the Standard Build.

Important to remember to select the deployment option that aligns with your project's requirements and hosting capabilities because that choice will impact how you deploy, maintain, and scale your Next.js application.