Next.js CLI

The Next.js CLI allows you to start, build, and export your application.

To get a list of the available CLI commands, run the following command inside your project directory:

Terminal

```
npx next -h
```

(npx comes with npm 5.2+ and higher)

The output should look like this:

Terminal

```
Usage
  $ next <command>

Available commands
  build, start, export, dev, lint, telemetry, info

Options
  --version, -v   Version number
  --help, -h      Displays this message

For more information run a command with the --help flag
  $ next build --help
```

You can pass any node arguments to `next` commands:

Terminal

```
NODE_OPTIONS='--throw-deprecation' next
NODE_OPTIONS='-r esm' next
NODE_OPTIONS='--inspect' next
```

Note: Running `next` without a command is the same as running `next dev`

Build

`next build` creates an optimized production build of your application. The output displays information about each route.

- **Size** – The number of assets downloaded when navigating to the page client-side. The size for each route only includes its dependencies.
- **First Load JS** – The number of assets downloaded when visiting the page from the server. The amount of JS shared by all is shown as a separate metric.

Both of these values are **compressed with gzip**. The first load is indicated by green, yellow, or red. Aim for green for performant applications.

You can enable production profiling for React with the `--profile` flag in `next build`. This requires Next.js 9.5↗:

Terminal

```
next build --profile
```

After that, you can use the profiler in the same way as you would in development.

You can enable more verbose build output with the `--debug` flag in `next build`. This requires Next.js 9.5.3:

Terminal

```
next build --debug
```

With this flag enabled additional build output like rewrites, redirects, and headers will be shown.

Development

`next dev` starts the application in development mode with hot-code reloading, error reporting, and more:

The application will start at `http://localhost:3000` by default. The default port can be changed with `-p`, like so:

Terminal

```
npx next dev -p 4000
```

Or using the `PORT` environment variable:

Terminal

```
PORT=4000 npx next dev
```

Note: `PORT` cannot be set in `.env` as booting up the HTTP server happens before any other code is initialized.

You can also set the hostname to be different from the default of `0.0.0.0`, this can be useful for making the application available for other devices on the network. The default hostname can be changed with `-H`, like so:

Terminal

```
npx next dev -H 192.168.1.2
```

Production

`next start` starts the application in production mode. The application should be compiled with `next build` first.

The application will start at `http://localhost:3000` by default. The default port can be changed with `-p`, like so:

Terminal

```
npx next start -p 4000
```

Or using the `PORT` environment variable:

Terminal

```
PORT=4000 npx next start
```

> **Note:** `PORT` cannot be set in `.env` as booting up the HTTP server happens before any other code is initialized.

> **Note:** `next start` cannot be used with `output: 'standalone'` or `output: 'export'`.

Keep Alive Timeout

When deploying Next.js behind a downstream proxy (e.g. a load-balancer like AWS ELB/ALB) it's important to configure Next's underlying HTTP server with keep-alive timeouts ↗ that are *larger* than the downstream proxy's timeouts. Otherwise, once a keep-alive timeout is reached for a given TCP connection, Node.js will immediately terminate that connection without notifying the downstream proxy. This results in a proxy error whenever it attempts to reuse a connection that Node.js has already terminated.

To configure the timeout values for the production Next.js server, pass `--keepAliveTimeout` (in milliseconds) to `next start`, like so:

Terminal

```
npx next start --keepAliveTimeout 70000
```

Lint

`next lint` runs ESLint for all files in the `pages/`, `app` (only if the experimental `appDir` feature is enabled), `components/`, `lib/`, and `src/` directories. It also provides a guided setup to install any required dependencies if ESLint is not already configured in your application.

If you have other directories that you would like to lint, you can specify them using the `--dir` flag:

Terminal

```
next lint --dir utils
```

Telemetry

Next.js collects **completely anonymous** telemetry data about general usage. Participation in this anonymous program is optional, and you may opt-out if you'd not like to share any information.

To learn more about Telemetry, please read this document↗.

Next Info

`next info` prints relevant details about the current system which can be used to report Next.js bugs. This information includes Operating System platform/arch/version, Binaries (Node.js, npm, Yarn, pnpm) and npm package versions (`next`, `react`, `react-dom`).

Running the following in your project's root directory:

Terminal

```
next info
```

will give you information like this example:

Terminal

```

    Operating System:
      Platform: linux
      Arch: x64
      Version: #22-Ubuntu SMP Fri Nov 5 13:21:36 UTC 2021
    Binaries:
      Node: 16.13.0
      npm: 8.1.0
      Yarn: 1.22.17
      pnpm: 6.24.2
    Relevant packages:
      next: 12.0.8
      react: 17.0.2
      react-dom: 17.0.2

```

This information should then be pasted into GitHub Issues.