Output File Tracing

During a build, Next.js will automatically trace each page and its dependencies to determine all of the files that are needed for deploying a production version of your application.

This feature helps reduce the size of deployments drastically. Previously, when deploying with Docker you would need to have all files from your package's `dependencies` installed to run `next start`. Starting with Next.js 12, you can leverage Output File Tracing in the `.next/` directory to only include the necessary files.

Furthermore, this removes the need for the deprecated `serverless` target which can cause various issues and also creates unnecessary duplication.

How It Works

During `next build`, Next.js will use `@vercel/nft` to statically analyze `import`, `require`, and `fs` usage to determine all files that a page might load.

Next.js' production server is also traced for its needed files and output at `.next/next-server.js.nft.json` which can be leveraged in production.

To leverage the `.nft.json` files emitted to the `.next` output directory, you can read the list of files in each trace which are relative to the `.nft.json` file and then copy them to your deployment location.

Automatically Copying Traced Files (experimental)

Next.js can automatically create a `standalone` folder which copies only the necessary files for a production deployment including select files in `node_modules`.

To leverage this automatic copying you can enable it in your `next.config.js`:

```
module.exports = {
  experimental: {
    outputStandalone: true,
  },
}
```

This will create a folder at `.next/standalone` which can then be deployed on it's own without installing `node_modules`.

Additionally, a minimal `server.js` file is also output which can be used instead of `next start`. This minimal server does not copy the `public` or `.next/static` folders by default as these should ideally be handled by a CDN instead, although these folders can be copied to the `standalone/public` and `standalone/.next/static` folders manually, after which `server.js` file will serve these automatically.

Caveats

- While tracing in monorepo setups, the project directory is used for tracing by default. For `next build packages/web-app`, `packages/web-app` would be the tracing root and any files outside of that folder will not be included. To include files outside of this folder you can set `experimental.outputFileTracingRoot` in your `next.config.js`.

```
// packages/web-app/next.config.js
module.exports = {
  experimental: {
    // this includes files from the monorepo base two directories up
    outputFileTracingRoot: path.join(__dirname, '../../'),
  },
}
```

- There are some cases that Next.js might fail to include required files, or might incorrectly include unused files. In those cases, you can export page configs props `unstable_includeFiles` and `unstable_excludeFiles` respectively. Each prop accepts an array of minimatch globs relative to the project's root to either include or exclude in the trace.
- Currently, Next.js does not do anything with the emitted `.nft.json` files. The files must be read by your deployment platform, for example Vercel, to create a minimal deployment. In a future release, a new command is planned to utilize these `.nft.json` files.