Runtime Configuration

> Generally you'll want to use build-time environment variables to provide your configuration. The reason for this is that runtime configuration adds rendering / initialization overhead and is incompatible with Automatic Static Optimization.

To add runtime configuration to your app open `next.config.js` and add the `publicRuntimeConfig` and `serverRuntimeConfig` configs:

```
module.exports = {
  serverRuntimeConfig: {
    // Will only be available on the server side
    mySecret: 'secret',
    secondSecret: process.env.SECOND_SECRET, // Pass through env variables
  },
  publicRuntimeConfig: {
    // Will be available on both server and client
    staticFolder: '/static',
  },
}
```

Place any server-only runtime config under `serverRuntimeConfig`.

Anything accessible to both client and server-side code should be under `publicRuntimeConfig`.

> A page that relies on `publicRuntimeConfig` **must** use `getInitialProps` to opt-out of Automatic Static Optimization. Runtime configuration won't be available to any page (or component in a page) without `getInitialProps`.

To get access to the runtime configs in your app use `next/config`, like so:

```
import getConfig from 'next/config'
import Image from 'next/image'

// Only holds serverRuntimeConfig and publicRuntimeConfig
const { serverRuntimeConfig, publicRuntimeConfig } = getConfig()
// Will only be available on the server-side
console.log(serverRuntimeConfig.mySecret)
// Will be available on both server-side and client-side
console.log(publicRuntimeConfig.staticFolder)

function MyImage() {
  return (
    <div>
      <Image
        src={`${publicRuntimeConfig.staticFolder}/logo.png`}
        alt="logo"
        layout="fill"
      />
    </div>
  )
}

export default MyImage
```

Related

Introduction to next.config.js: Learn more about the configuration file used by Next.js.

Environment Variables: Access environment variables in your Next.js application at build time.