Gatsby has built-in support for loading environment variables into the browser and Functions. Loading environment variables into Node.js requires a small code snippet.

In development, Gatsby will load environment variables from a file named `.env.development`. For builds, it will load from `.env.production`.

A `.env` file could look like:

```
GATSBY_API_URL=https://dev.example.com/api
API_KEY=927349872349798
```

To load these into Node.js, add the following code snippet to the top of your `gatsby-config.js` file:

```
require("dotenv").config({
  path: `.env.${process.env.NODE_ENV}`,
})
```

This loads `process.env.GATSBY_API_URL` and `process.env.API_KEY` for use in `gatsby-*.js` files and functions.

For example, when configuring a plugin in `gatsby-config.js`:

```
require("dotenv").config({
  path: `.env.${process.env.NODE_ENV}`,
})

module.exports = {
  plugins: [
    {
      resolve: `gatsby-source-custom`,
      options: {
        apiKey: process.env.API_KEY,
      },
    },
  ],
}
```

Accessing Environment Variables in the browser.

By default, environment variables are only available in Node.js code and are not available in the browser as some variables should be kept secret and not exposed to anyone visiting the site.

To expose a variable in the browser, you must preface its name with `GATSBY_`. So `GATSBY_API_URL` will be available in browser code but `API_KEY` will not.

Variables are set when JavaScript is compiled so when the development server is started or you build your site.

```
import React, { useState, useEffect } from "react"

function App() {
```

```
  const [data, setData] = useState()

  useEffect(async () => {
    const result = await fetch(
      `${process.env.GATSBY_API_URL}/users`
    ).then(res => res.json())

    setData(result.data)
  })

  return (
    <ul>
      {data.map(user => (
        <li key={user.id}>
          <a href={user.url}>{user.name}</a>
        </li>
      ))}
    </ul>
  )
}

export default App
```

Add `.env*` files to .gitignore

Environment variable files should not be committed to Git as they often contain secrets which are not safe to add to Git. Instead, add `.env.*` to your `.gitignore` file and set up the environment variables manually on Gatsby Cloud and locally.

Environment variables on Gatsby Cloud

In Gatsby Cloud you can configure environment variables in each site's "Site Settings". You can read more about managing environment variables in Gatsby Cloud and environment variables specific to Gatsby Cloud.

Additional Environments (Staging, Test, etc.)

You can create additional environments beyond `development` and `production` through customizing `dotenv`'s `path` configuration. E.g. to add a staging environment you could run the Gatsby build command like:

```
STAGING=true gatsby build
```

and then in your `gatsby-config.js` file:

```
if (process.env.STAGING) {
  require("dotenv").config({
    path: `.env.${process.env.NODE_ENV}.staging`,
  })
} else {
  require("dotenv").config({
    path: `.env.${process.env.NODE_ENV}`,
```

```
  })
}
```

Reserved Environment Variables:

You can not override certain environment variables that are used internally:

- `NODE_ENV`
- `PUBLIC_DIR`

Gatsby also allows you to specify another environment variable when running the local development server (e.g. `npm run develop`):

- `ENABLE_GATSBY_REFRESH_ENDPOINT`

This allows you to refresh your sourced content. See Refreshing Content.

Gatsby detects an optimal level of CPU cores to use during `gatsby build` based on the OS reported number of physical CPUs. For builds that are run in virtual environments, you may need to adjust the number of worker parallelism with the `GATSBY_CPU_COUNT` environment variable. See Multi-core builds.