Resources

If you're not familiar with Gatsby's lifecycle, see the overview Gatsby Lifecycle APIs.

Proxying API requests in development

People often serve the frontend React app from the same host and port as their backend implementation.

To tell the development server to proxy any unknown requests to your API server in development, add a `proxy` field to your `gatsby-config.js`, for example:

```
module.exports = {
  proxy: {
    prefix: "/api",
    url: "http://dev-mysite.com",
  },
}
```

or:

```
module.exports = {
  proxy: [
    {
      prefix: "/api",
      url: "http://dev-mysite.com",
    },
    {
      prefix: "/api2",
      url: "http://dev2-mysite.com",
    },
  ],
}
```

This way, when you `fetch('/api/todos')` in development, the development server will recognize that it's not a static asset, and will proxy your request to `http://dev-mysite.com/api/todos` as a fallback.

Keep in mind that `proxy` only has effect in development (with `gatsby develop`), and it is up to you to ensure that URLs like `/api/todos` point to the right place in production.

Advanced proxying

Sometimes you need more granular/flexible access to the development server. Gatsby exposes the Express.js development server to your site's `gatsby-config.js` where you can add Express middleware as needed.

```
const { createProxyMiddleware } = require("http-proxy-middleware") //v1.x.x
// Use implicit require for v0.x.x of 'http-proxy-middleware'
// const proxy = require('http-proxy-middleware')
// be sure to replace 'createProxyMiddleware' with 'proxy' where applicable

module.exports = {
```

```
  developMiddleware: app => {
    app.use(
      "/.netlify/functions/",
      createProxyMiddleware({
        target: "http://localhost:9000",
        pathRewrite: {
          "/.netlify/functions/": "",
        },
      })
    )
  },
}
```

Keep in mind that middleware only has effect in development (with `gatsby develop`).

Self-signed certificates

If you proxy to local APIs with self-signed certificates, set the option `secure` to `false`.

```
const { createProxyMiddleware } = require("http-proxy-middleware") //v1.x.x
// Use implicit require for v0.x.x of 'http-proxy-middleware'
// const proxy = require('http-proxy-middleware')
// be sure to replace 'createProxyMiddleware' with 'proxy' where applicable

module.exports = {
  developMiddleware: app => {
    app.use(
      "/.netlify/functions/",
      createProxyMiddleware({
        target: "http://localhost:9000",
        secure: false, // Do not reject self-signed certificates.
        pathRewrite: {
          "/.netlify/functions/": "",
        },
      })
    )
  },
}
```