This guide will help you understand how to set up Gatsby to pull data from Flotiq.

Flotiq is a headless CMS with a primary focus on developer experience and integration capabilities. Flotiq's cloud-based dashboard allows you to design your content types and work with your data, but Flotiq also provides support for powerful integrations. One of Flotiq's key principles is to provide an effortless way to consume your content in the applications you build. It's solved by supporting technologies like OpenAPI 3.0 or Zapier as well as by providing customized API docs and SDK libraries based on your content types.

Using a website generator like Gatsby to consume content stored in Flotiq is one of the most frequent use cases. The integration is enabled by the gatsby-source-flotiq source plugin, which bridges the dynamic REST API of your Flotiq account with Gatsby's GraphQL.

> *This guide will reference the gatsby-starter-blog blog starter, but remember to check out the more advanced starters prepared to work with Flotiq:*
>
> - *Recipe website Gatsby starter*
> - *Event calendar Gatsby starter*
> - *Project portfolio Gatsby starter*
> - *Gatsby starter blog*
> - *Gatsby and Snipcart boilerplate, sourcing products from Flotiq*
> - *Gatsby and Snipcart, e-commerce with products and categories from Flotiq*

Setup

Quickstart using a Flotiq starter

1. **Start project from template using Gatsby CLI**

   ```
   gatsby new my-blog-starter https://github.com/flotiq/gatsby-starter-blog
   ```

2. **Set up "Blog Post" Content Type in Flotiq**

   Create your Flotiq.com account. Next, create the `Blog Post` Content Type straight from the `Type definitions` page.

   Note: You can also define the `Blog Post` Content Type using the Flotiq REST API.

3. **Configure application**

   The next step is to configure your application to know from where it has to fetch the data.

   You need to create a file called `.env` inside the root project directory, with the following structure. The Read-Only API key can be copied from the Flotiq user settings menu, under API Keys.

   ```
   GATSBY_FLOTIQ_BASE_URL=https://api.flotiq.com
   GATSBY_FLOTIQ_API_KEY=YOUR FLOTIQ API KEY
   ```

4. **Start developing**

   Navigate into your new site's directory and start it up.

   ```
   cd my-blog-starter/
   npm install
   ```

```
gatsby develop
```

If you wish to import example blog posts to your account, before running `gatsby develop` run this Node.js script provided by Flotiq.

Note: You need to put your Full Access API key in `.env` for import to work, but this can be swapped back to the Read Only API key afterwards. You don't need the Blog Post content type already in your account. If you already have posts with ids `blogpost-1` and `blogpost-2` they will be overwritten.

```
node ./.flotiq/importExample.js
```

It will add 1 image and 2 blog posts to your Flotiq account.

Setting up the Flotiq source plugin in a Gatsby project

The Blog starter you've just set up uses the gatsby-source-flotiq source plugin to pull content from Flotiq.

Here are the steps to use this source plugin in other Gatsby projects:

> *If you're using the provided starter - all the following steps have been already taken care of, you can dive into the relevant project files to verify how it's been done.*

1. Install the plugin:

   ```
   npm install gatsby-source-flotiq
   ```

2. Provide API credentials in `.env` (see above for more details)

3. Register the source plugin in `gatsby-config.js`: Make sure your `gatsby-config.js` contains the following configuration:

   ```
   // required to pull the variables from .env
   require("dotenv").config()

   module.exports = {
     // ...
     plugins: [
       {
         resolve: "gatsby-source-flotiq",
         options: {
           baseUrl: process.env.GATSBY_FLOTIQ_BASE_URL,
           authToken: process.env.GATSBY_FLOTIQ_API_KEY,
         },
       },
     ],
     // ...
   }
   ```

By default the source plugin will pull all the data you've stored in Flotiq. In some cases, e.g. when building sites that have thousands of pages - you'll likely want to limit the amount of pulled data during development. You can do that either by narrowing down the scope to specific content types or by limiting the number of pulled objects (see the

`includeTypes` and `objectLimit` options). The source plugin will do its best to resolve any dependencies for you.

> *You can find more details about how to configure the Flotiq source plugin in the plugin's README.*

Accessing Flotiq data in Gatsby

Once you finish configuring your environment you can start developing. The source plugin will pull all the content from your Flotiq account and create respective Gatsby GraphQL nodes. It will also preserve the relations you've setup in Flotiq, so the GraphQL nodes will be automatically linked.

For example, if you define a relation between `BlogPost` and `Category` content types in Flotiq - they will retain their relationship in Gatsby's GraphQL, so you can retrieve that in a query:

```
query {
  allBlogPost {
    nodes {
      id
      slug
      category {
        name
      }
    }
  }
}
```

The next step is implementing these queries in your `gatsby-node.js` file:

```
const path = require(`path`)
const { createFilePath } = require(`gatsby-source-filesystem`)
exports.createPages = async ({ graphql, actions }) => {
  const { createPage } = actions

  const blogPost = path.resolve(`./src/templates/blog-post.js`)
  const result = await graphql(`
    query GetBlogPosts {
      allBlogpost(sort: { fields: flotiqInternal___createdAt, order: DESC }) {
        edges {
          node {
            headerImage {
              extension
              id
            }
            content
            id
            slug
            title
          }
        }
      }
    }
```

```
  `)

  if (result.errors) {
    throw result.errors
  }

  // Create blog posts pages.
  const posts = result.data.allBlogpost.edges
  posts.forEach((post, index) => {
    const previous = index === posts.length - 1 ? null : posts[index + 1].node
    const next = index === 0 ? null : posts[index - 1].node

    createPage({
      path: post.node.slug,
      component: blogPost,
      context: {
        slug: post.node.slug,
        previous,
        next,
      },
    })
  })
}

exports.onCreateNode = ({ node, actions, getNode }) => {
  const { createNodeField } = actions

  if (node.internal.type === `MarkdownRemark`) {
    const value = createFilePath({ node, getNode })
    createNodeField({
      name: `slug`,
      node,
      value,
    })
  }
}
```

Now you'll want to let Gatsby create appropriate pages for your Content Objects. This example uses a `blogPost` component as a template:

```
import React from "react"
import { graphql } from "gatsby"

class BlogPostTemplate extends React.Component {
  render() {
    const post = this.props.data.blogpost
    const siteTitle = this.props.data.site.siteMetadata.title
    const { previous, next } = this.props.pageContext

    return (
      <article>
```

```
        <header>
          <h1>{post.title}</h1>
        </header>
        {post.headerImage && post.headerImage[0] && (
          <img
            src=
{`${process.env.GATSBY_FLOTIQ_BASE_URL}/image/1920x0/${post.headerImage[0].id}.${post.h

            alt="${post.title}"
            style={{ maxWidth: "100%", height: "auto" }}
          />
        )}
        <div dangerouslySetInnerHTML={{ __html: post.content }} />
      </article>
    )
  }
}

export default BlogPostTemplate

export const pageQuery = graphql`
  query BlogPostBySlug($slug: String!) {
    site {
      siteMetadata {
        title
      }
    }
    blogpost(slug: { eq: $slug }) {
      id
      title
      content
      headerImage {
        extension
        id
      }
    }
  }
`
```

Deployment

Once you're finished with your website there are several ways to deploy it. You can use any hosting provider you choose - Netlify, Heroku, AWS S3, Cloudflare, etc.

If you're using Gatsby Cloud, you can use the Flotiq integration to streamline your workflow by providing live updates in preview and push-button deployments.

You can read the relevant Flotiq Gatsby Cloud integration documentation page to learn more.

Summary

This guide has gone through the key points of setting up a Gatsby starter, configuring it to work with Flotiq, and kicking off your development process. Remember to check out other Flotiq Gatsby starters already prepared to kick-start your next project. You can also join the Flotiq Discord channel if you need any help!