Creating a new site from a starter

This is the recommended way to build new sites so that you start with good best practises and don't need to re-implement everything yourself.

Setting up WordPress

If you don't have a WP site yet, we recommend creating a local WP instance using Local by Flywheel but as long as you have a WP instance hosted somewhere you're good to get started.

Log into your WP instance and navigate to `/wp-admin/plugin-install.php`. Once you're there, search for and install these two plugins:

- WPGraphQL
- WPGatsby

We need WPGraphQL so that we have an efficient and flexible way for `gatsby-source-wordpress` to pull data into Gatsby. WPGatsby is also required as it sets up a WP admin event log that Gatsby will use to pull content changes into Gatsby after our initial build. It also offers some additional features like Gatsby Preview and webhooks to kick off builds on your build service when content changes in WP.

Setting up Gatsby

First you'll need to ensure you have npm, Gatsby, and Nodejs installed on your computer. Follow the GatsbyJS guide on setting up your development environment.

Setting up `gatsby-source-wordpress`

You're super close to being able to run your new Gatsby/WP site now!

1. In a terminal, `cd` (navigate) to the directory you want your Gatsby site to live in.
2. To create a new site using the official WP Gatsby starter, run `gatsby new my-wordpress-gatsby-site https://github.com/gatsbyjs/gatsby-starter-wordpress-blog`. That will fetch this starter and put it in a new directory called `my-wordpress-gatsby-site` and run `npm install`.

Now that you have a local Gatsby site installed, open it in your favourite IDE or text editor and open up the `gatsby-config.js` file. At the top of the `plugins` array you'll see the following code:

```
{
  /**
   * First up is the WordPress source plugin that connects Gatsby
   * to your WordPress site.
```

```
      *
      * visit the plugin docs to learn more
      * https://github.com/gatsbyjs/gatsby/blob/master/packages/gatsby-source-wordpress/RE
      *
      */
    resolve: `gatsby-source-wordpress`,
    options: {
      // the only required plugin option for WordPress is the GraphQL url.
      url:
        process.env.WPGRAPHQL_URL ||
        `https://wpgatsbydemo.wpengine.com/graphql`,
    },
  },
```

As you can see, the `url` option is the only required option. You will want to replace this with the URL of the `/graphql` endpoint you added by installing WPGraphQL in your WP instance. Out of the box WPGraphQL will add the GraphQL endpoint at `http://[yoursite.com]/graphql`.

```
{
  resolve: `gatsby-source-wordpress`,
  options: {
    url: `https://demo.wpgraphql.com/graphql`,
  },
},
```

Running the site

Now that you have the source plugin configured you can check it out by `cd`ing into your site directory in a terminal and running `gatsby develop`. You should see the plugin output some lines in your terminal showing which types of data it's fetching from WPGraphQL. On the first build (`gatsby develop` or `gatsby build`) `gatsby-source-wordpress` will fetch all available public data from WPGraphQL. On subsequent builds, it will only fetch changed data to keep your builds quick.

Open http://localhost:8000 in the browser to view your site.

Modifying the starter code

For a guide on how different parts of the code work, check out this gatsbyjs.com guide on using WordPress data.

Helpful plugin options for larger sites

If you have a larger site you may find that the amount of time it takes for you to run `gatsby develop` is too long. Any time you add a new npm package, edit

your `gatsby-config.js` or `gatsby-node.js` files, Gatsby will clear the cache and the source plugin will need to re-fetch all data. We can mitigate that with a couple options:

Limiting the amount of data fetched with the `options.type.[typename].limit` option

```
{
  resolve: `gatsby-source-wordpress`,
  options: {
    url: `https://demo.wpgraphql.com/graphql`,
    type: {
        __all: {
            limit: process.env.NODE_ENV === `development` ? 50 : null
        }
    }
  },
},
```

This option will limit the total number of nodes fetched per-type. In this example we're using the special `__all` type which applies to all node types. If you wanted to change the limit based on a specific type you could do something like this:

```
{
  resolve: `gatsby-source-wordpress`,
  options: {
    url: `https://demo.wpgraphql.com/graphql`,
    type: {
        Post: {
            limit: 50
        },
        Page: {
            limit: 50
        }
    }
  },
},
```

Note that `process.env.NODE_ENV === "development" ? 50 : null` will apply this limit only during development while `limit: 50` will apply a limit during development and in production builds.

Hard caching MediaItem local files in development using the `options.develop.hardCacheMediaFiles` option

When the Gatsby cache is cleared all image files that the source plugin fetched from WordPress will be deleted. That means they will then need to be immediately re-downloaded when you run `gatsby develop` again. We can eliminate

this problem in development with the `hardCacheMediaFiles` option:

```
{
  resolve: `gatsby-source-wordpress`,
  options: {
    url: `https://demo.wpgraphql.com/graphql`,
    develop: {
        hardCacheMediaFiles: true,
    }
  },
},
```

Now you'll only ever need to fetch media item files 1 time on your local machine.