GitLab Pages are very similar to GitHub Pages. GitLab Pages also supports custom domain names and SSL certificates and includes a continuous integration platform.

Create a new GitLab repository, create a new Gatsby site if you haven't already, and add the GitLab remote.

```
git init
git remote add origin git@gitlab.com:examplerepository
git add .
git push -u origin master
```

You can deploy sites on GitLab Pages with or without a custom domain. If you choose to use the default setup (without a custom domain), or if you create a project site, you will need to set up your site with path prefixing. If adding a custom domain, you can skip the Path Prefix step, and remove `--prefix-paths` from the `.gitlab-ci.yml` file.

Add Path Prefix to Gatsby

As the site will be hosted under yourname.gitlab.io/examplerepository/, you will need to configure Gatsby to use the Path Prefix plugin.

In the `gatsby-config.js`, set the `pathPrefix` to be added to your site's link paths. The `pathPrefix` should be the project name in your repository. (ex. `https://gitlab.com/yourname/examplerepository/` - your `pathPrefix` should be `/examplerepository`). See the docs page on path prefixing for more.

```
module.exports = {
  pathPrefix: `/examplerepository`,
}
```

Build and deploy with GitLab CI

To use GitLab's continuous integration (CI), you need to add a `.gitlab-ci.yml` configuration file. This is the file that GitLab uses to manage the CI job.

The online editor on the GitLab website contains a pre-built template for Gatsby deployment.

To use the template open your repository on their website, select the 'Setup CI/CD' option on the center menu, and it will create a new blank `.gitlab-ci.yml` for you. Now select the 'Apply a GitLab CI YAML Template' drop-down, and type 'Gatsby' into the filter. Select the Gatsby option, click 'Commit Changes', and you are done!

If adding this manually to your project, the file needs to contain a few required fields:

```
image: node:latest

# This folder is cached between builds
# https://docs.gitlab.com/ce/ci/yaml/README.html#cache
cache:
  paths:
    - node_modules/
    # Enables git-lab CI caching. Both .cache and public must be cached, otherwise
builds will fail.
    - .cache/
```

```
    - public/

pages:
  script:
    - npm install
    - ./node_modules/.bin/gatsby build --prefix-paths
  artifacts:
    paths:
      - public
  only:
    - master
```

The CI platform uses Docker images/containers, so `image: node:latest` tells the CI to use the latest node image. `cache:` caches the `node_modules` folder in between builds, so subsequent builds should be a lot faster as it doesn't have to reinstall all the dependencies required. `pages:` is the name of the CI stage. You can have multiple stages, e.g. 'Test', 'Build', 'Deploy' etc. `script:` starts the next part of the CI stage, telling it to start running the below scripts inside the image selected. `npm install` and `./node_modules/.bin/gatsby build --prefix-paths` will install all dependencies, and start the static site build, respectively.

`./node_modules/.bin/gatsby build --prefix-paths` was used so you don't have to install gatsby-cli to build the image, as it has already been included and installed with `npm install`. `--prefix-paths` was used because *without* that flag, Gatsby ignores your pathPrefix. `artifacts:` and `paths:` are used to tell GitLab Pages where the static files are kept. `only:` and `master` tells the CI to only run the above instructions when the master branch is deployed.

Add that configuration, and with the next master branch push, your site should have been built correctly. This can be checked by going to your repository on GitLab, and selecting CI/CD in the sidebar. This will then show you a log of all jobs that have either succeeded or failed. You can click on the failed status, and then select the job to get more information about why your build may have failed.

If all went well, you should now be able to access your site. It will be hosted under gitlab.io - for example if you have a repository under your namespace, the url will be yourname.gitlab.io/examplerepository.

Visit the GitLab Pages to learn how to set up custom domains and find out about advanced configurations.