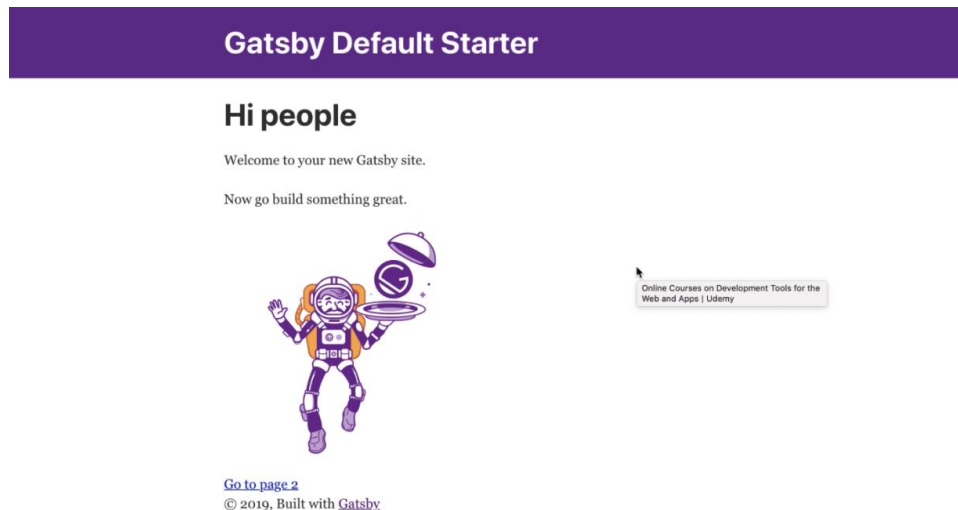


## Variables de Entorno Predefinidas

En esta ocasión, veremos cómo podemos utilizar variables de entorno predefinidas al utilizar GitLab CI.

Imaginemos que tenemos un sitio web muy sencillo:



Supongamos que hacemos un cambio en este sitio web, pero es un cambio sencillo que no se ve reflejado en la interfaz de usuario, como un texto, una imagen o algo así. Por lo tanto, no tenemos forma de saber a simple vista qué versión estamos usando, o qué versión se desplegó. Una idea para solucionar esto es agregar un poco de información extra, como el número de versión, con lo cual podremos saber qué commit ha generado esta versión en particular de nuestro sitio web.

Afortunadamente, GitLab ya viene con un montón de variables de entorno predefinidas que podemos utilizar y que podemos ver en esta página: [https://docs.gitlab.com/ee/ci/variables/predefined\\_variables.html](https://docs.gitlab.com/ee/ci/variables/predefined_variables.html)

**Predefined variables reference** All tiers All offerings

Predefined [CI/CD variables](#) are available in every GitLab CI/CD pipeline.

Some variables are only available with more recent versions of [GitLab Runner](#).

You can [output the values of all variables available for a job](#) with a `script` command.

There are also a number of [variables you can use to configure runner behavior](#) globally or for individual jobs.

ⓘ You should avoid [overriding](#) predefined variables, as it can cause the pipeline to behave unexpectedly.

| Variable                  | GitLab | Runner | Description   |
|---------------------------|--------|--------|---|
| <code>CHAT_CHANNEL</code> | 10.6   | all    | The Source chat channel that triggered the <a href="#">ChatOps</a> command.               |
| <code>CHAT_INPUT</code>   | 10.6   | all    | The additional arguments passed with the <a href="#">ChatOps</a> command.                 |
| <code>CHAT_USER_ID</code> | 14.4   | all    | The chat service's user ID of the user who triggered the <a href="#">ChatOps</a> command. |
| <code>CI</code>           | all    | 0.4    | Available for all jobs executed in CI/CD. <code>true</code> when available.               |

Una forma fácil de incluir la versión de nuestro sitio web es incluir la versión corta del “commit hash”, y existe la variable predefinida llamada `CI_COMMIT_SHORT_SHA`:

|                                  |      |     |  |
|----------------------------------|------|-----|--|
| <code>CI_COMMIT_SHORT_SHA</code> | 11.7 | all | The first eight characters of <code>CI_COMMIT_SHA</code> . |
|----------------------------------|------|-----|--|

Esta variable nos dará los primeros 8 caracteres del commit hash:



Así que estos 8 dígitos es suficiente para identificar qué commit ha generado la versión del sitio web que se esté usando.

Usar variables de entorno predefinidas proporcionadas por GitLab es una forma bastante fácil de hacer cosas muy dinámicas en nuestros pipelines.

Así que, sean cuales sean nuestras necesidades, sin duda merece la pena echar un vistazo a la referencia de las variables para ver cuáles de ellas son interesantes para nuestros casos.

Pero para lo que necesitamos en este momento, esto va a ser suficiente. Así que echemos un vistazo a cómo podemos usar las variables en nuestros scripts de nuestro pipeline.

Para agregar la versión al compilado de nuestro sitio web, tenemos que ver nuestro Job “build website”, ya que aquí es donde se está construyendo y es donde se está publicando el artifact que contiene nuestro sitio web.

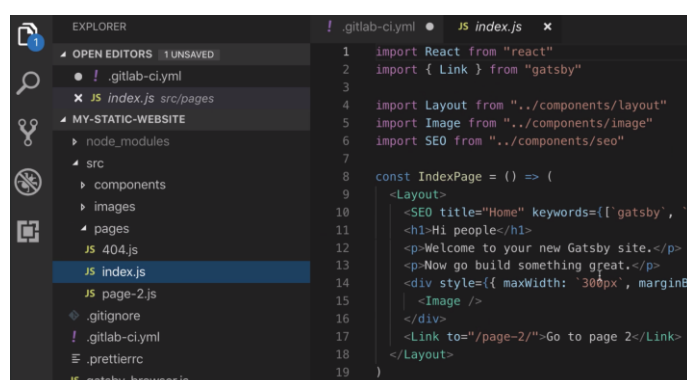
Así que, lo que haremos primero es un simple “echo” con el nombre de la variable de entorno que queremos:

```

9   build website:
10     stage: build
11     script:
12       - echo $CI_COMMIT_SHORT_SHA
13       - npm install
14       - npm install -g gatsby-cli
15       - gatsby build
16     artifacts:
17       paths:
18         - ./public
19

```

Cuando el script se ejecute, se imprimirá el valor de la variable de entorno `CI_COMMIT_SHORT_SHA`. Pero aun así, esto no nos ayudará a incluir el hash en nuestro archivo html. Para eso, vamos a abrir el archivo `index.js`:

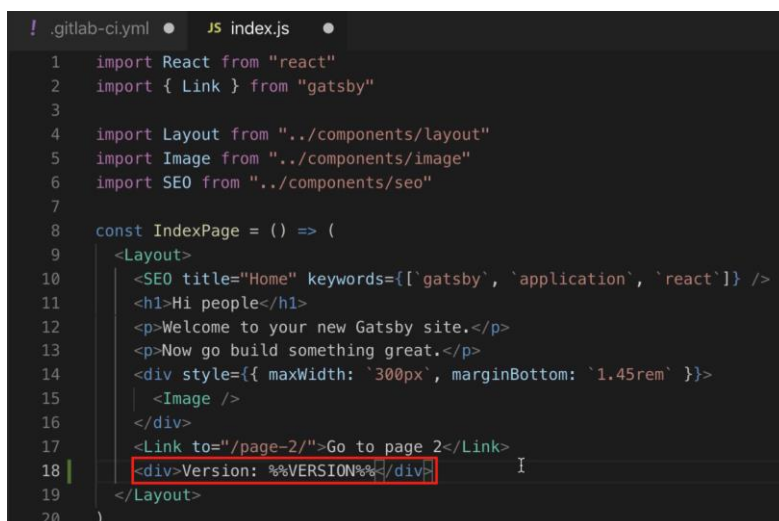


```

1  import React from "react"
2  import { Link } from "gatsby"
3
4  import Layout from "../components/layout"
5  import Image from "../components/image"
6  import SEO from "../components/seo"
7
8  const IndexPage = () => (
9    <Layout>
10      <SEO title="Home" keywords={['gatsby', 'application', 'react']} />
11      <h1>Hi people</h1>
12      <p>Welcome to your new Gatsby site.</p>
13      <p>Now go build something great.</p>
14      <div style={{ maxWidth: `300px`, marginBottom: `300px` }}>
15        <Image />
16      </div>
17      <Link to="/page-2/">Go to page 2</Link>
18    </Layout>
19  )

```

Ahora, la idea es la siguiente: insertaremos dentro del archivo `index.js`, en la parte del código html, un marcador, y vamos a tratar de encontrar una manera de reemplazar ese marcador cada vez que construyamos el sitio web. Por lo tanto, dentro del repositorio no tenemos que hacer ningún commit, pero este marcador se reemplazará cada vez que ejecutemos el pipeline. Así que vamos a añadir un simple div con el contenido `"%%Version: VERSION%%"`:



```

1  import React from "react"
2  import { Link } from "gatsby"
3
4  import Layout from "../components/layout"
5  import Image from "../components/image"
6  import SEO from "../components/seo"
7
8  const IndexPage = () => (
9    <Layout>
10      <SEO title="Home" keywords={['gatsby', 'application', 'react']} />
11      <h1>Hi people</h1>
12      <p>Welcome to your new Gatsby site.</p>
13      <p>Now go build something great.</p>
14      <div style={{ maxWidth: `300px`, marginBottom: `1.45rem` }}>
15        <Image />
16      </div>
17      <Link to="/page-2/">Go to page 2</Link>
18      <div>Version: %%VERSION%%</div>
19    </Layout>
20  )

```

Para reemplazar el marcador por el commit hash, utilizaremos el comando `"sed"`. Así que este paso debe ejecutarse después de que Gatsby haya terminado de construir el sitio web. En nuestro caso, el comando se vería así:

```

build website:
  stage: build
  script:
    - echo $CI_COMMIT_SHORT_SHA
    - npm install
    - npm install -g gatsby-cli
    - gatsby build
    - sed -i "s/%%VERSION%%/$CI_COMMIT_SHORT_SHA/" ./public/index.html
  artifacts:
    paths:
      - ./public

```

Entonces, utilizamos nuestro marcador %%VERSION%%, seguido del nombre de la variable de entorno CI\_COMMIT\_SHORT\_SHA y luego el nombre del archivo donde queremos hacer el reemplazo del texto.

Una de las cosas que podemos notar cuando se ejecuta el Job “build website” es que el comando echo imprime el commit hash corto:

```

Using docker image sha256:5a401340b79fc623c9aec2a679f16ceb8a3b8865446691564bb104394fd0ce20 for no
Running on runner-fa6cab46-project-11731229-concurrent-0 via runner-fa6cab46-srm-1558489342-fe3d2
Initialized empty Git repository in /builds/vdespa/my-static-website/.git/
Fetching changes...
Created fresh repository.
From https://gitlab.com/vdespa/my-static-website
* [new branch]      master    -> origin/master
Checking out 6529e4ed as master...

Skipping Git submodules setup
$ echo $CI_COMMIT_SHORT_SHA
6529e4ed
$ npm install

> sharp@0.21.3 install /builds/vdespa/my-static-website/node_modules/sharp
> (node install/libvips && node install/dll-copy && prebuild-install) || (node-gyp rebuild && no

```

Y cuando finaliza el pipeline, si volvemos a nuestro sitio web y actualizamos la página, podemos ver que el mismo commit hash estará incluido en el html:

[Go to page 2](#)

Version: 6529e4ed

© 2019, Built with [Gatsby](#)

---