GIT HUB

GitHuB

Por ahora todo lo que venía ocurriendo en Git era de manera local, no necesitábamos nada de internet para guardar nuestros commits y nuestro repositorio.
Ahora **queremos compartir nuestro trabajo con otros** (compañeros de proyecto, clientes,etc) ¡Para eso utilizamos

Github!

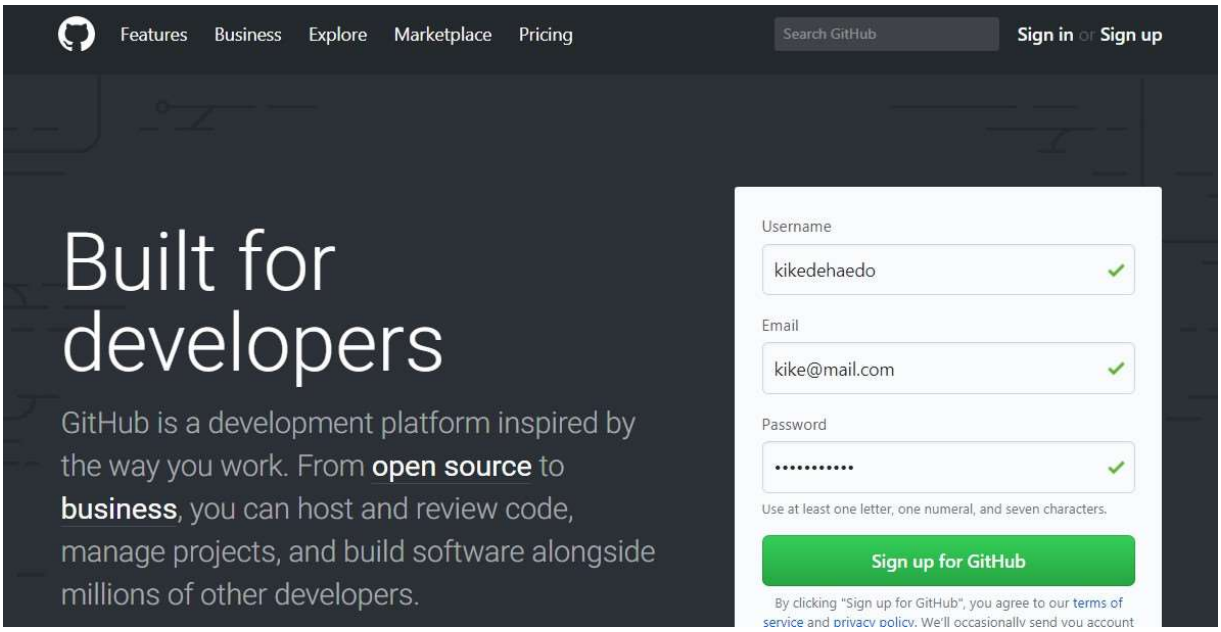
https://github.com/

"GitHub is a development platform inspired by the way you work. From open source to business, you can host and review code, manage projects, and build software alongside 50 million developers.

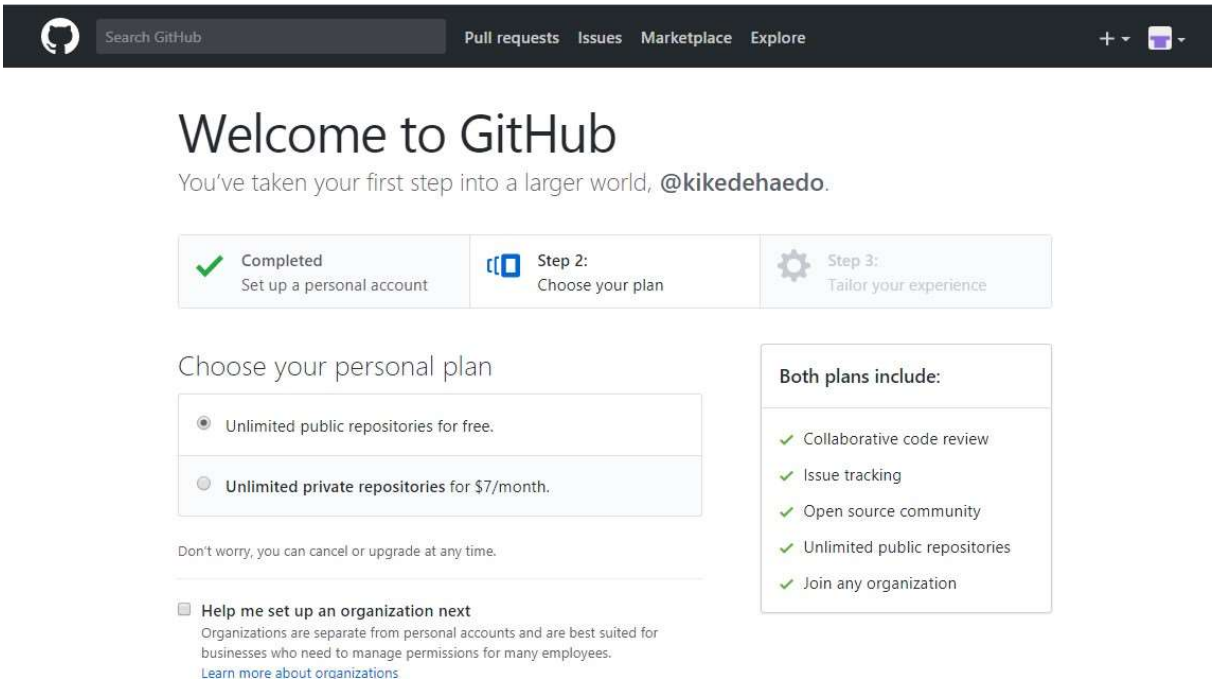
GitHub es una plataforma de desarrollo colaborativo dedicada a alojar proyectos utilizando el sistema de control de versiones Git

Utiliza el framework Ruby on Rails desarrollado por GitHub, Inc. (anteriormente conocida como Logical Awesome). El código de los proyectos alojados en GitHub se almacena típicamente de forma pública, aunque utilizando una cuenta de pago, también permite hospedar repositorios privados.

Creando una cuenta en GitHuB

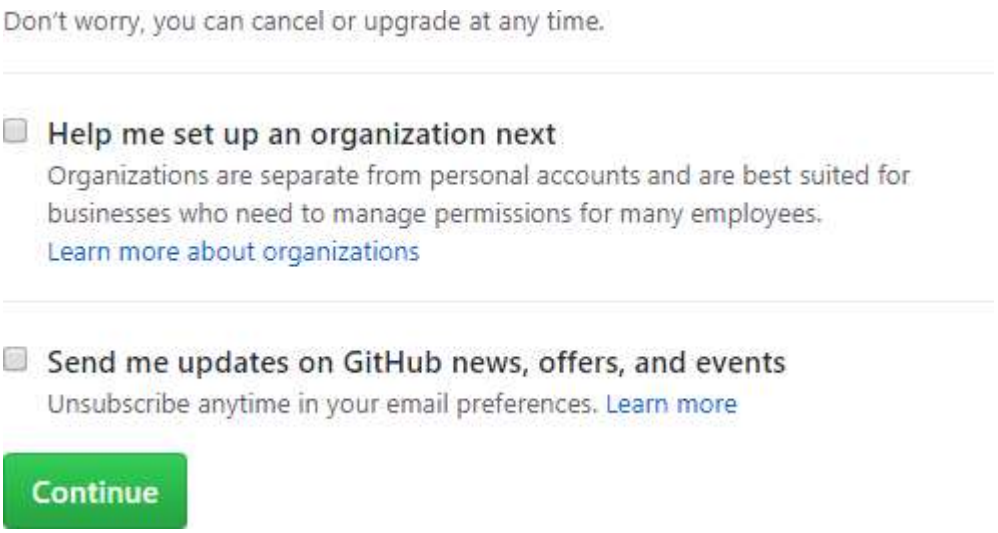


Entra a la página de registro de GitHub https://github.com/ y desde allí crea tu cuenta. Recomendable usar el mismo email que usamos en nuestro perfil de Git anteriormente.

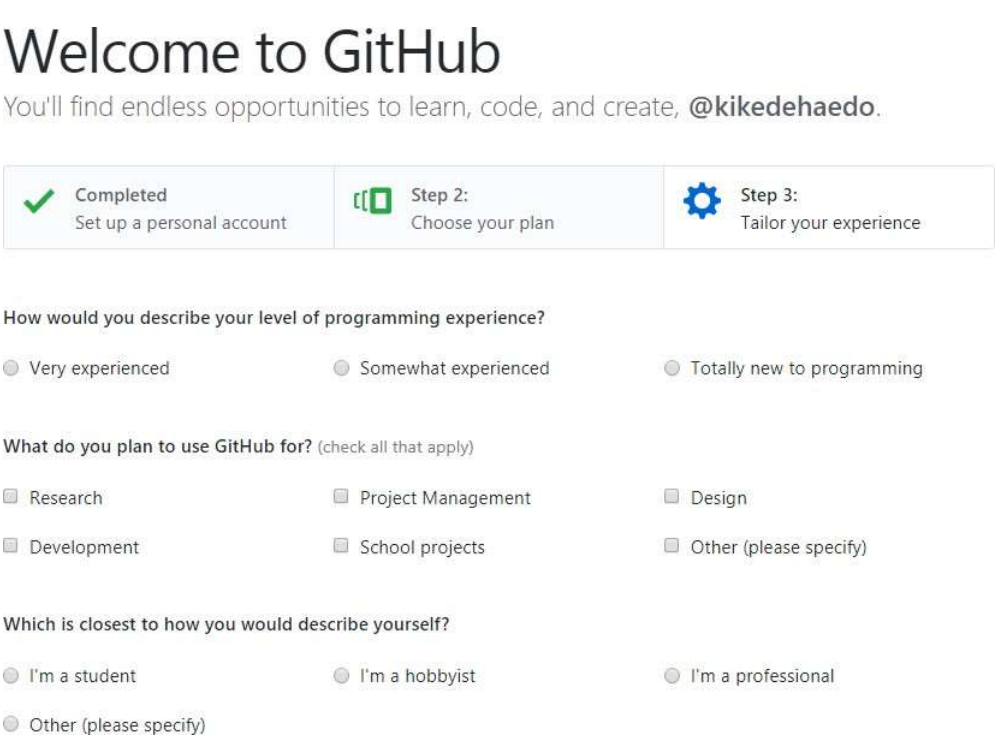


Lo siguiente es configurar si queremos que la cuenta sea gratuita o paga. ¿Diferencias? La más singular es que si nos vamos por el plan gratuito no vamos a poder crear

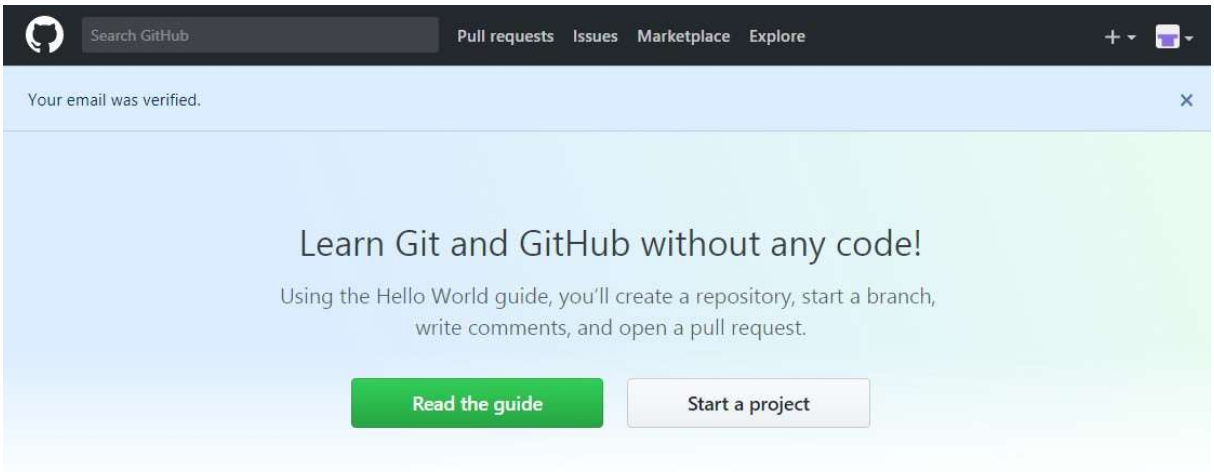
Repositorios Privados, o sea nuestro código va a estar disponible para todos los usuarios de GitHub. Por el momento usemos una cuenta gratuita.



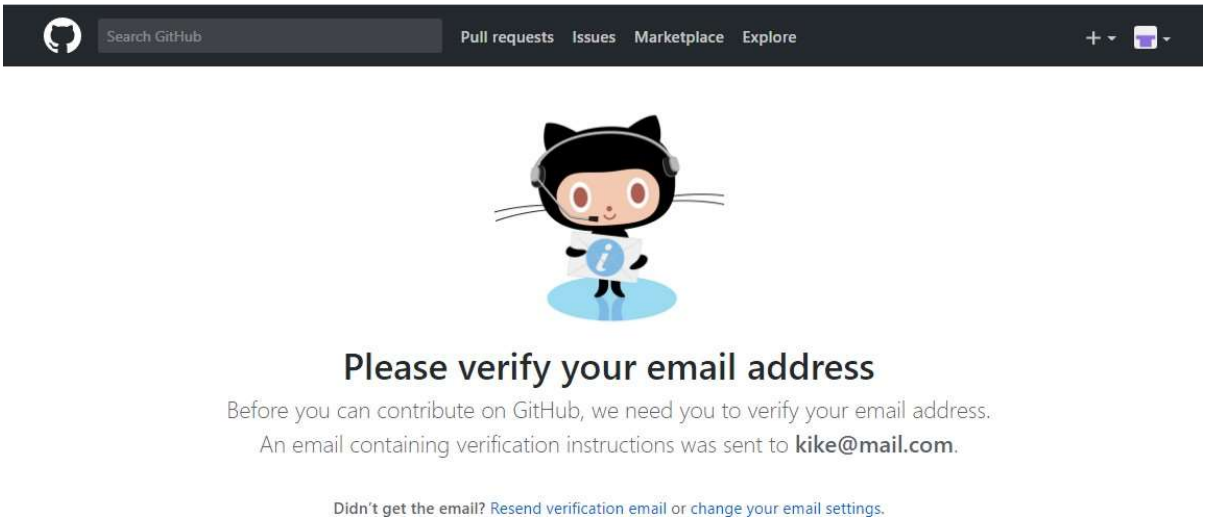
Luego podríamos pedirle ayuda a Github para la creación de algún tipo de organización, y si queremos que GitHub nos envíe información (¡newsletters alert!) a nuestro correo.



La tercera pantalla nos pide información acerca de nosotros, nuestro nivel de desarrolladores, para qué usaremos GitHub, entre otros.



Luego de esta pantalla de confirmación nos pedirá que validemos nuestra casilla de correo electrónico siguiendo las instrucciones que nos lleguen a la misma.

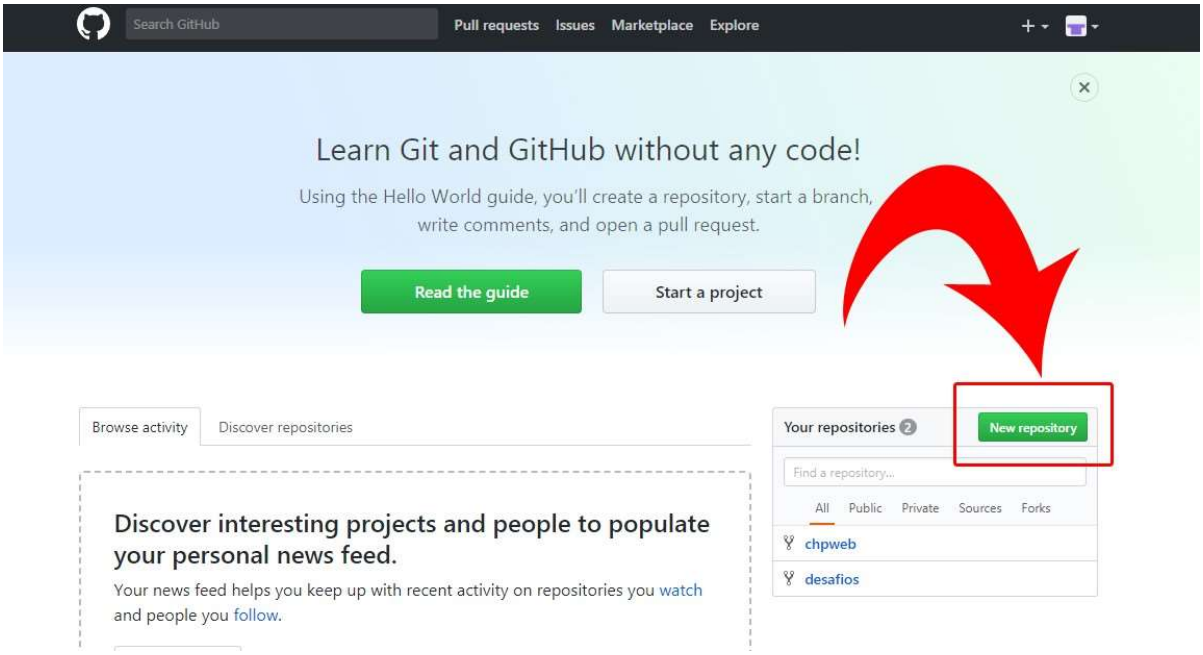


¡Y listo! Ya estamos listos para empezar a crear nuestro propio repositorio online.

Git clone (Clonando repoSitorioS remotoS)

Lo primero que vamos a hacer es crear un nuevo repositorio en GitHub.

Hacemos click en Nuevo Repositorio

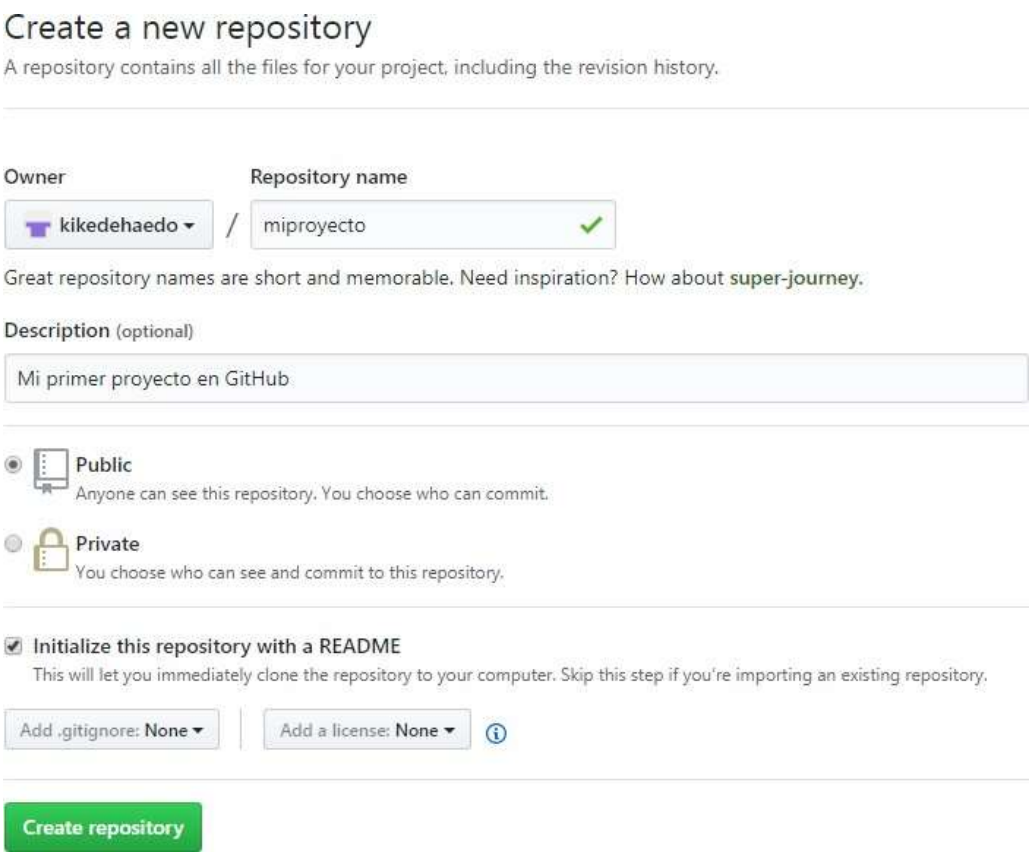


En la próxima pantalla **debemos ingresar el nombre de nuestro repositorio** , asignarleuna descripción, podríamos (opcional) cambiar nuestro repositorio de Público a Privadoy podemos adicionar un “readme” o un .gitignore (abordaremos este último un poco más adelante en esta misma unidad)

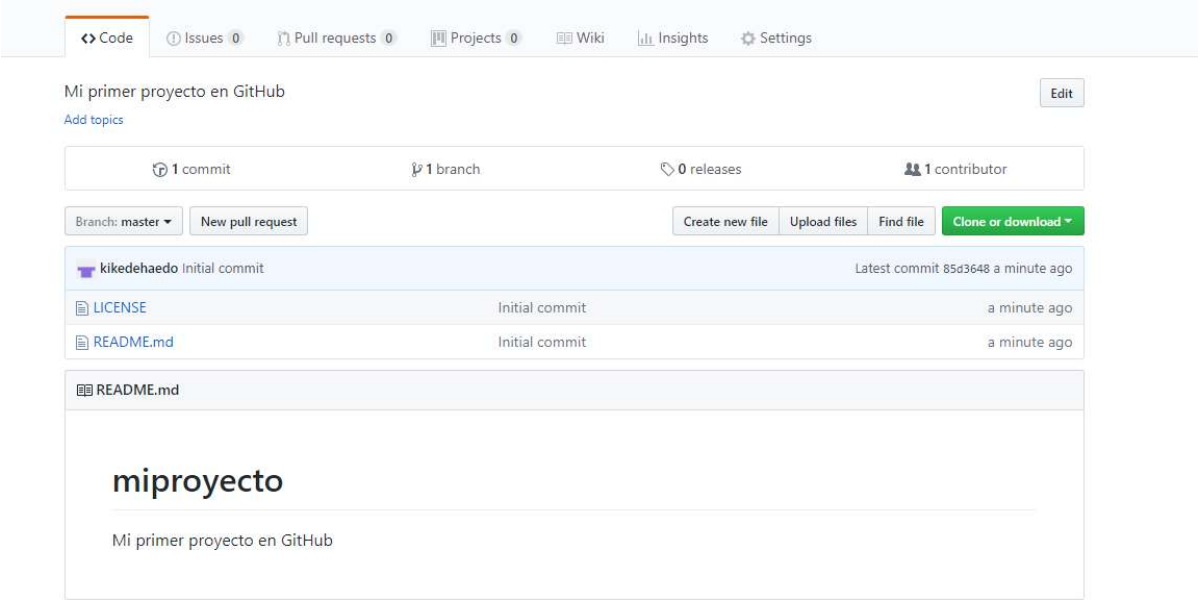
README

El Readme podría ser el primer archivo de nuestro repositorio en donde le contemos a la comunidad de GitHub**de qué trata nuestro proyecto**
También podemos adjudicar algún tipo de licencia a nuestro proyecto. Si quieres saber más sobre licencias:
https://es.wikipedia.org/wiki/Licencia_de_software
https://choosealicense.com/

Luego de ingresar esa información dar click en Crear repositorio (Create repository)

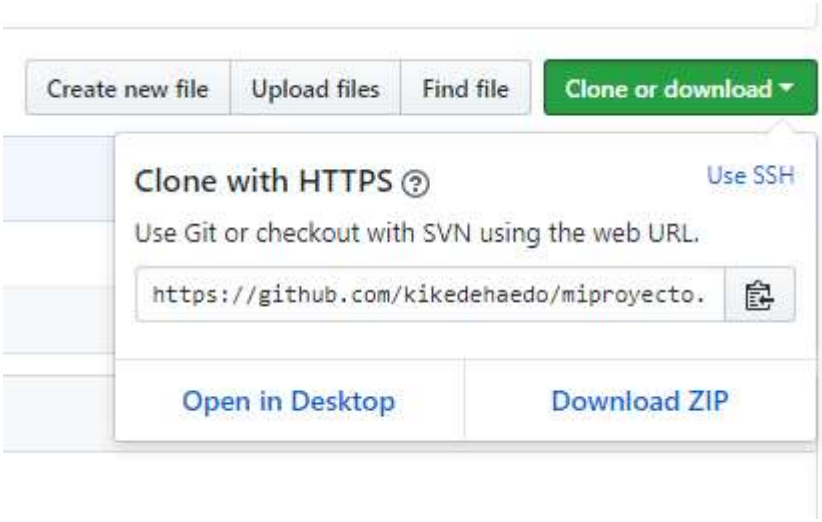


Una vez creado vamos a encontrar el detalle de nuestro nuevo repositorio



En la parte superior derecha encontraremos un botón verde, el cual nos va a dar las siguientes opciones:

-Si queremos clonar este repositorio
-Si solamente queremos descargarlo en un archivo Zip.



Para clonarlo tenemos dos opciones:
-HTTPS
-SSH
Elijamos por el momento la versión HTTPS, debemos copiar la URL que nos proporcione, en este caso: https://github.com/kikedehaedo/miproyecto.git

Una vez copiado este enlace, volvamos a nuestra terminal y con el comando *git clone + [LA URL QUE COPIAMOS DE NUESTRO REPOSITORIO]* clonamos los archivos del proyecto dentro de la carpeta, en este caso, llamada: "miproyecto" :

```
john@MyShopSolutions MINGW64 /c/
$ git clone https://github.com/kikedehaedo/miproyecto.git
Cloning into 'miproyecto'...
remote: Counting objects: 4, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (4/4), done.
```

Si listamos los archivos en consola vamos a ver los archivos de nuestro repositorio:

```
john@MyShopSolutions MINGW64 /c/miproyecto (master)
$ dir
LICENCE README.md
```

Llave SSH

Git utiliza la autenticación a través de claves públicas SSH
sistema ha de generarse una, si es que aún no la tiene. El proceso para hacerlo es similar en casi cualquier sistema operativo.

Ya creaste tu cuenta de Github, y tienes tu usuario de Git y tu e-mail (los mismo que los de GitHub) Ahora tienes que lograr **una comunicación entre GitHub y tu entorno local**. Para ello existen las llaves SSH que se generan desde la terminal:

```
john@MyShopSolutions MINGW64 /c/miproyecto (master)
$ ssh-keygen -t rsa -b 4096 -C "mimail@mail.com"
```

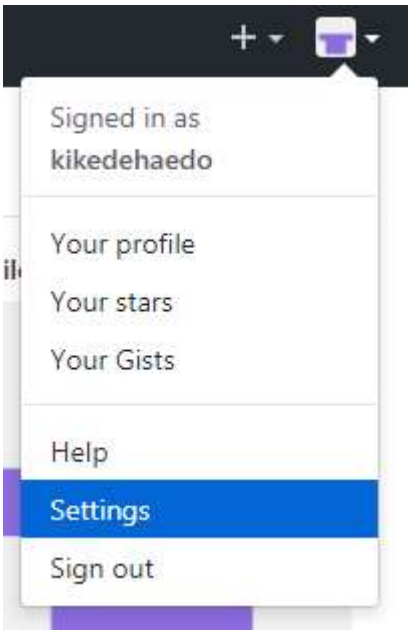
Luego de crearla debemos de copiarla dentro de nuestra configuración de GitHub. Para conseguirla vamos a usar el siguiente comando:

```
john@MyShopSolutions MINGW64 /c/miproyecto (master)
$ cat ~/.ssh/id_rsa.pub
```

O en MAC:

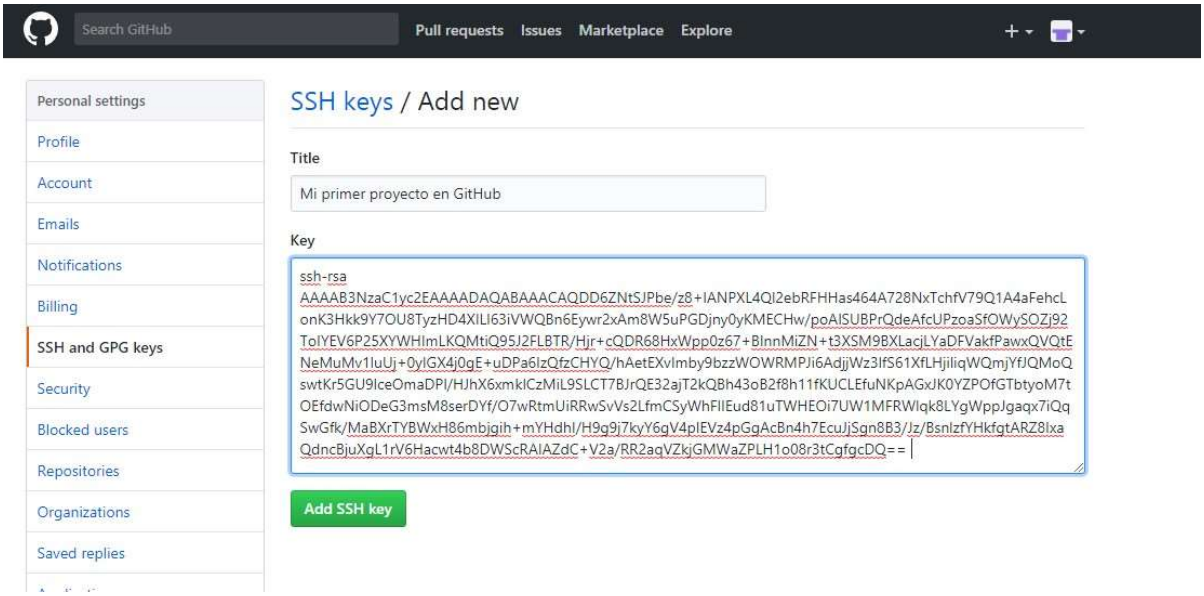
```
john@MyShopSolutions MINGW64 /c/miproyecto (master)
$ pbcopy < ~/.ssh/id_rsa.pub
```

Una vez copiada la llave en el clip-board de nuestro ordenador, tenemos que ingresar el código de la llave a nuestra configuración de perfil en GitHub:

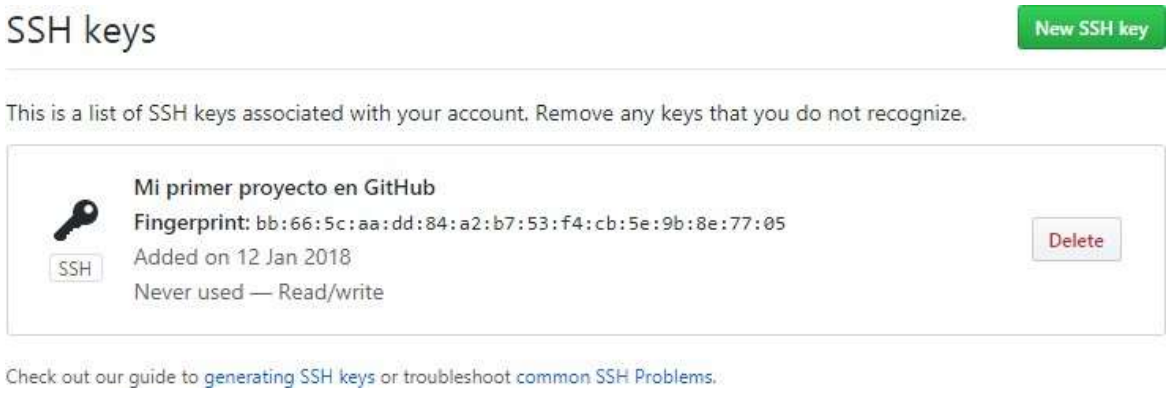


Una vez en settings vamos a la barra de herramientas de la izquierda y seleccionamos “SSH and GPG keys”

Aparecerá la lista de llaves asociadas a nuestros proyectos, en este caso vamos a agregar una nueva, agregando un nombre y copiando el código de esa llave.



Una vez agregada (click en “Add SSH key”) nos llegará un mail confirmando el proceso y la siguiente pantalla:



Una llave para enlazar Github con tu ordenador (una sola vez hasta que la formatees)

Git remote

Si queremos **generar un enlace entre un repositorio local y un repositorio remoto**

Debemos hacerlo por del comando *git remote*

Vamos a vincular nuestro repositorio local "nuevo_repo" a nuestro nuevo repositorio en gitHub

```
john@MyShopSolutions MINGW64 /c/git/nuevo_repo (master)
$ git remote add origin git@github.com:kikedehaedo/miproyecto.git
```

El comando es: git remoto [NOMBRE POR CONVENCION] [URL DE NUESTRO REPOSITORIO REMOTO]

En resumen:
git remote add [origin] [SSH/HTTPS] Conecta un repositorio con nuestro equipo local.
git remote -v Lista las conexiones existentes.
git remote remove [origin] Elimina una conexión con algún repositorio.

Como buenas prácticas, se recomienda lo siguiente:

- El nombre del repositorio creado en Github (u otra plataforma) debe ser el mismo que el nombre de la carpeta del proyecto.
- Primero crear el repositorio remoto en Github, luego clonarlo al repositorio local en tu máquina y de ahí empezar a trabajar.

Git pull - Git fetch

Ahora vamos a manejar los cambios para que se vean respaldados en GitHub.
Lo primero que vamos a hacer es **traernos archivos desde el repositorio remoto en GitHub**. Para eso vamos a usar el comando *git fetch*

```
john@MyShopSolutions MINGW64 /c/git/nuevo_repo (master)
$ git fetch origin master
remote: Counting objects: 10, done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 10 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (10/10), done.
From github.com:kikedehaedo/miproyecto
 * branch            master     -> FETCH_HEAD
 * [new branch]      master     -> origin/master
```

Le decimos a nuestra terminal que se traiga los archivos del repositorio llamado "origin" y de la rama "master"

Nos ha creado una nueva rama (origin/master), no se han mezclado, ahora tenemos que **mezclar los cambios que están en esta nueva rama**, y para eso usamos el ya conocido git merge

```
john@MyShopSolutions MINGW64 /c/git/nuevo_repo (master)
$ git merge origin/master
```

Ahora los cambios (nuevos archivos, cambios en archivos, etc) se han fusionado con nuestros archivos locales.

En caso de tener algún tipo de error al momento de fusionar recuerda que puedes agregar al comando *git merge…* lo siguiente siguiente:

```
john@MyShopSolutions MINGW64 /c/git/nuevo_repo (master)
$ git merge origin/master --allow-unrelated-histories
```

Este anexo a nuestro comando *merge* básicamente **permite la fusión cuando se niega a fusionar historias que no comparten un antecesor común**

Esta opción se puede utilizar para anular esta seguridad al fusionar historias de dos proyectos que comenzaron susvidas de forma independiente. Como es una ocasión muy rara, no existe ninguna variable de configuración para habilitar esto por defecto y no se agregará por el momento.

¿Qué tal si hubiera una forma que hiciera estas dos cosas al mismo tiempo?
Para ello existe el comando *git pull*

```
john@MyShopSolutions MINGW64 /c/git/nuevo_repo (master)
$ git pull origin master
```

Luego nos aparece en la terminal **confirmación de este merge**

¡y listo! ya tenemos loscambios en nuestro repositorio Local. Todo con un solo comando en la terminal.

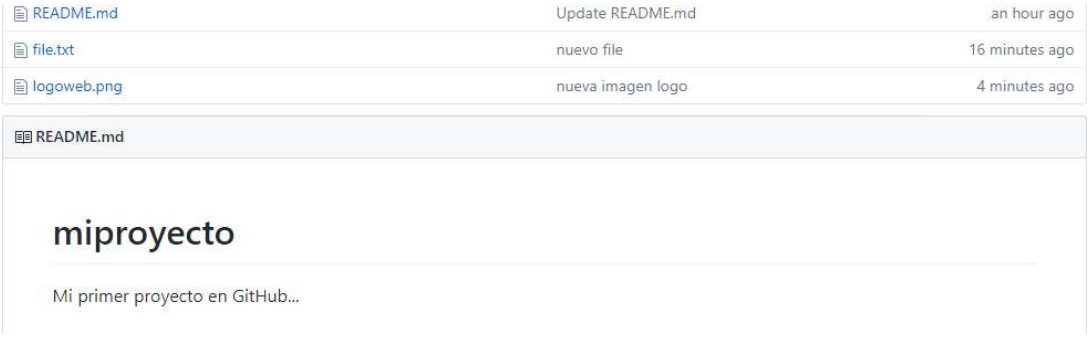
Git PuSh

Ahora vamos a **enviar nuevos cambios a nuestro repositorio remoto**

Para ello vamos a usar un comando llamado *git push*, para empujar nuevos cambios. En este caso tenemosun nuevo archivo llamado: "logoweb.png"

```
john@MyShopSolutions MINGW64 /c/git/nuevo_repo (master)
$ git push origin master
Counting objects: 8, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (7/7), done.
Writing objects: 100% (8/8), 32.50 KiB | 6.50 MiB/s, done.
Total 8 (delta 3), reused 0 (delta 0)
remote: Resolving deltas: 100% (3/3), completed with 1 local object.
To github.com:kikedehaedo/miproyecto.git
  545883f..4705b35  master -> master
```

Si chequeamos en nuestra página de GitHub vamos a encontrar este nuevo archivo en la rama master del proyecto.



Podemos enviar otras ramas:

```
john@MyShopSolutions MINGW64 /c/git/nuevo_repo (master)
$ git pull responsive
```

De esta manera enviamos la rama llamada “responsive” anteriormente creada con el comando *git branch*

.gitignore (Ignorando archivoS no deSeadoS)

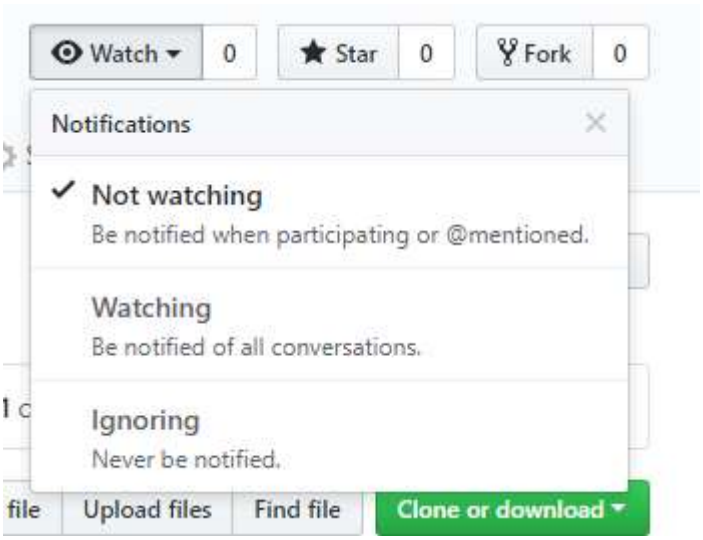
En el archivo *.gitignore* incluyes archivos y carpetas que no quieras o no necesitas subir, como por ejemplo archivos donde tengas llaves de servicios como aws, datos de bases de datos y toda información sensible. También incluyes carpetas como node_modules,ya que no es necesario subirla (cuando descargas el proyecto en otro lado usas el package.json para hacer la instalación), y más...

Para eso debemos crear el archivos *.gitignore,* y abriéndolo con un editor de texto agregarle el nombre de los archivos (nombre.extensión) que queramos ignorar.
Los archivos no serán subidos al repositorio remoto, solo estarán en el repositorio local.

MAS PROPIEDADES DE GITHUB

En la parte superior podríamos seguir o dejar de seguir al proyecto en donde estemos situados: tenemos las opciones Watching/Watch, Ignoring o Not Watching (Ver/Viendo, Ignorando, No Verlo)

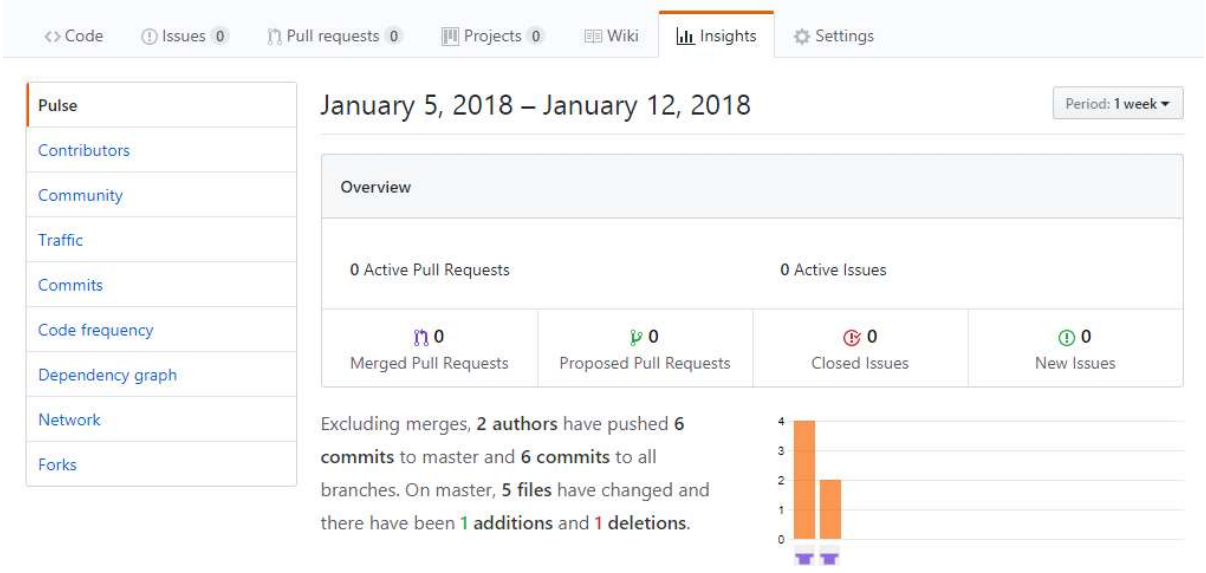
En caso de que nos interesen los avances del proyecto, situarlo en Watching para así poder ver los avances del mismo en nuestra página de inicio de GitHub.



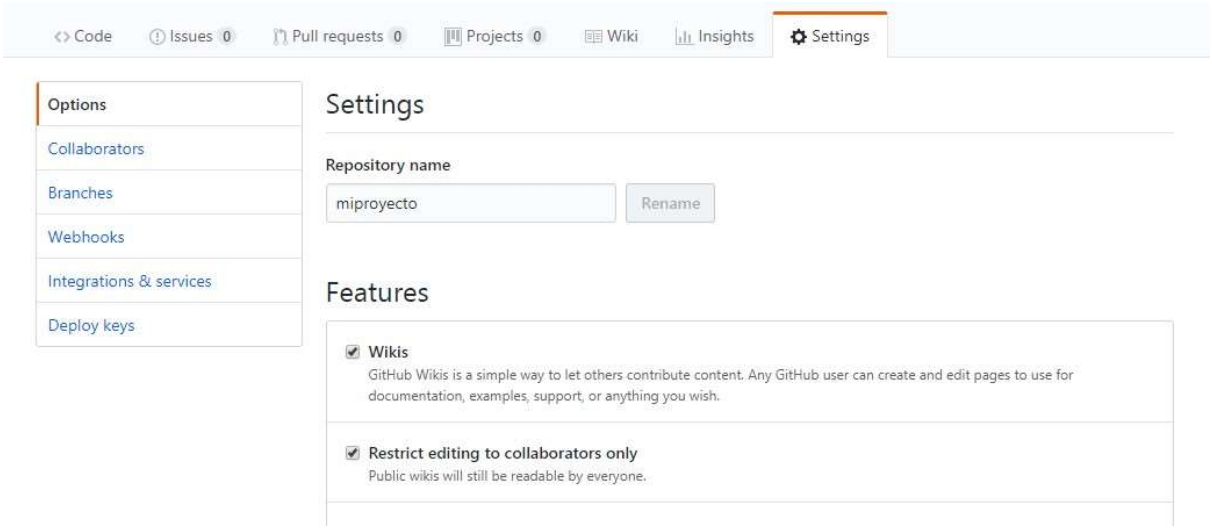
También podríamos “darle like” (Star) al proyecto

Y como última opción podríamos hacerle un Fork al proyecto (en caso de que sea un proyecto de otro usuario) y copiarlo. Este comando nos va a clonar el proyecto y crear un nuevo proyecto en nuestro GitHub.

En "Insights” podemos ver las estadísticas de nuestro proyecto:

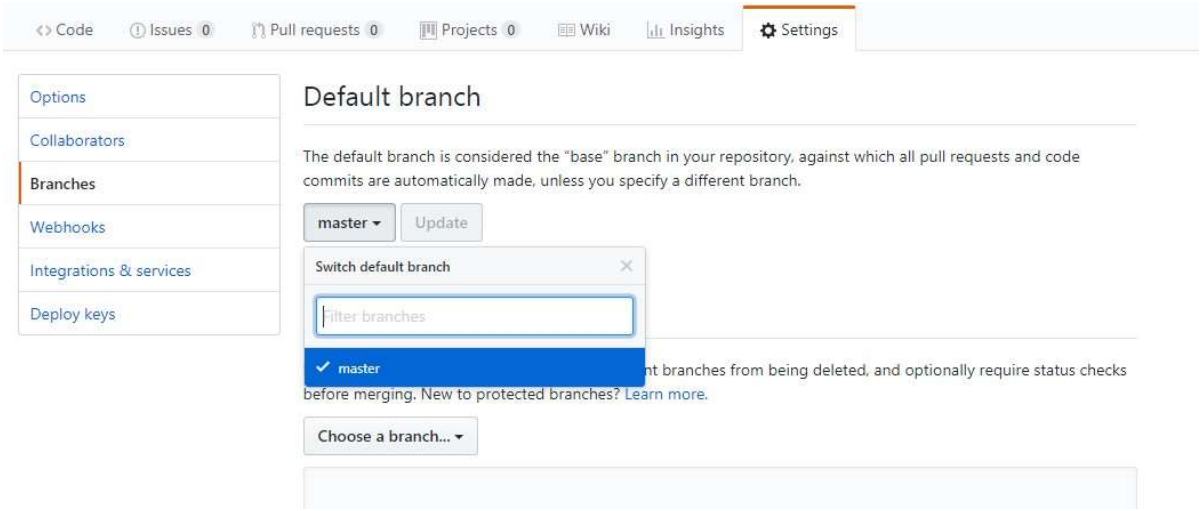


En los Settings de nuestro repositorio podemos configurar distintos aspectos muy interesantes



Por ejemplo, en la solapa “Collaborators” **podríamos sumar a otro usuario para que nos ayude con nuestro proyecto** y empiece a colaborar con commits sobre nuestros distintos documentos.

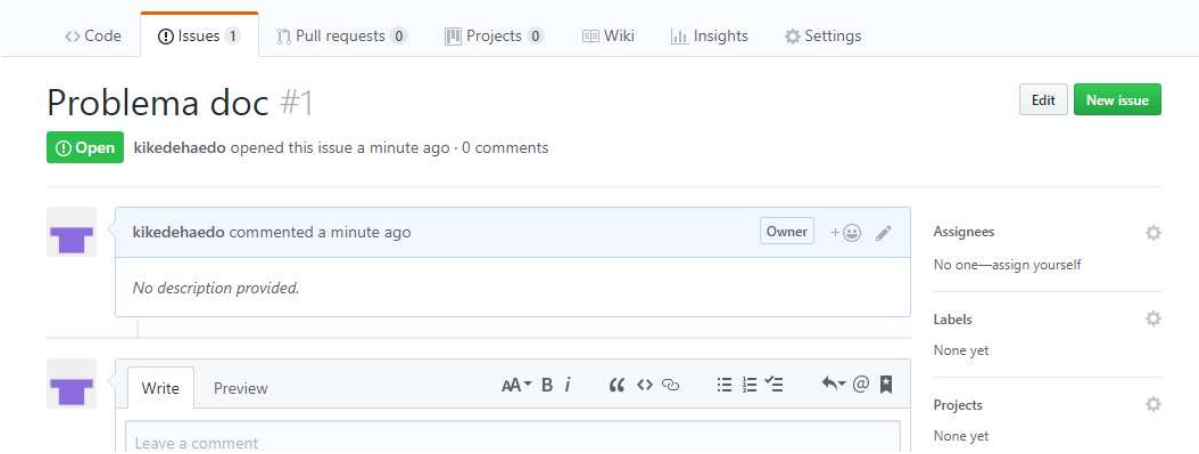
Otra opción a remarcar es la llamada “Branches”, para cambiar cuál sería nuestra rama principal



Y también podríamos **proteger a nuestra rama “master”**
de esa manera al protegerlamediante un “pull request” cada actualización que afecte la rama maestra queda a disposición del administrador del proyecto a ser cotejada y autorizada.

En la solapa "Issues” los diferentes colaboradores del proyecto **pueden remarcar y**

alertar acerca de diferentes "bugs" o "problemas" dentro del proyecto.

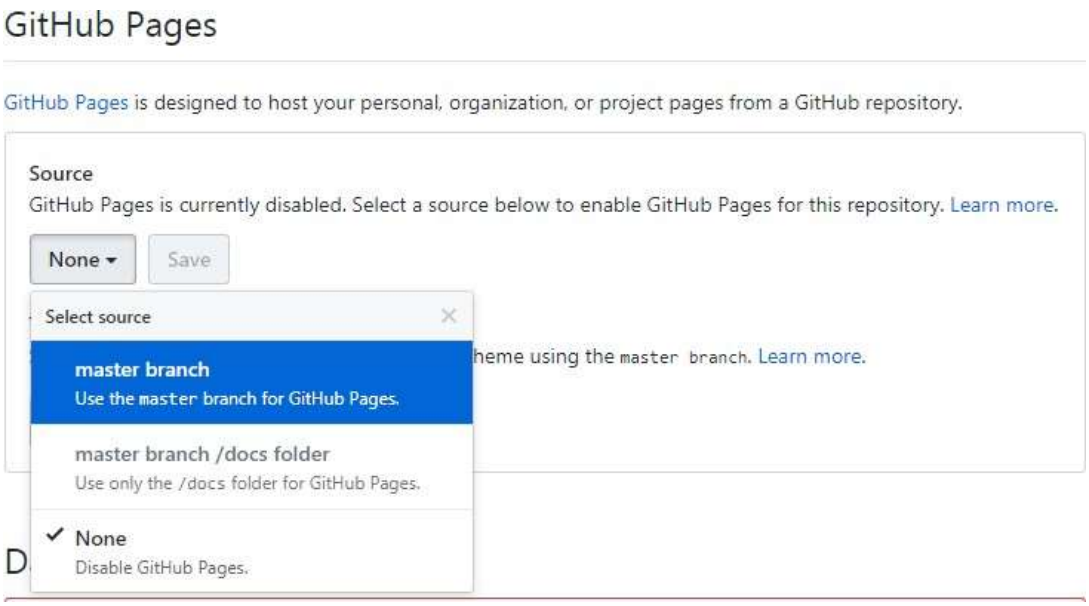


Para enterarte de todas las características de GitHub puedes visitar el siguiente link: https://help.github.com/

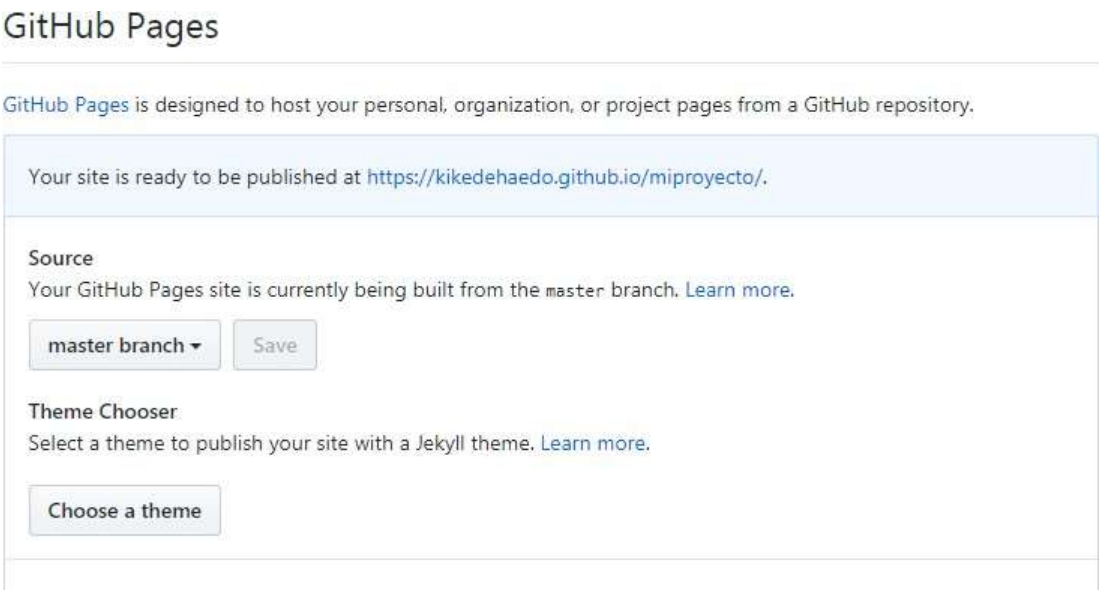
GitHuB PageS

GitHub nos **permite publicar nuestros proyectos online**. Para generar un GitHub page debemos ir a los "Settings" de nuestro repositorio y activar nuestro GitHub page, seleccionar qué rama queremos usar, guardamos los cambios y GitHub cumplirá la función básica de cualquier otro Hosting.

Importante: El proyecto solo de archivos estáticos, ningún archivo que requiera de BackEnd especial.



Luego de seleccionar y salvar los cambios la página recargara y nos mostrará cuál es la URL de nuestro proyecto


GitHub Pages
GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.
Your site is ready to be published at https://kikedehaedo.github.io/miproyecto/.
Source
Your GitHub Pages site is currently being built from the master branch. Learn more.
master branch
Save
Theme Chooser
Select a theme to publish your site with a Jekyll theme. Learn more.
Choose a theme