

GIT



Git es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia, la confiabilidad y compatibilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente.

Instalación

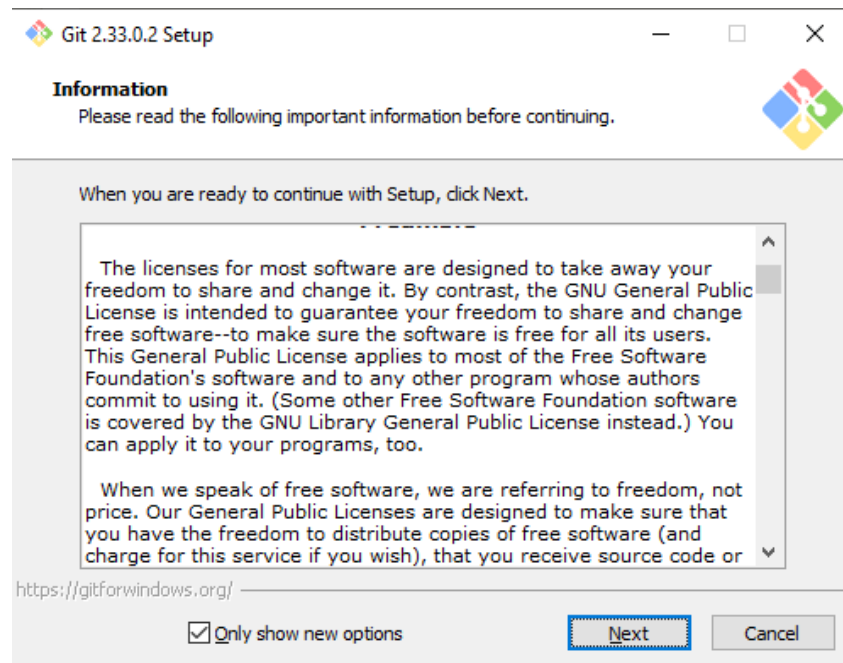
La instalación de git se realizará en la siguiente página web:

<https://git-scm.com/>



The screenshot shows the Git website homepage. At the top left is the Git logo and the tagline "--everything-is-local". A search bar is on the top right. The main content area describes Git as a "free and open source distributed version control system" and lists its features: "easy to learn", "tiny footprint", "lightning fast performance", "cheap local branching", "convenient staging areas", and "multiple workflows". To the right of this text is a diagram showing a branching model with stacks of code blocks connected by colored lines. Below the main text are four sections: "About" (advantages of Git), "Documentation" (command reference, Pro Git book, etc.), "Downloads" (GUI clients, binary releases), and "Community" (bug reporting, mailing list, etc.). On the right side, there is a monitor displaying the "Latest source Release 2.33.0" and a "Download for Windows" button.

Nos descargamos la versión que mejor se adapte a nuestro ordenador, al igual que la configuración que elegiremos que esté más acorde a nuestras necesidades.

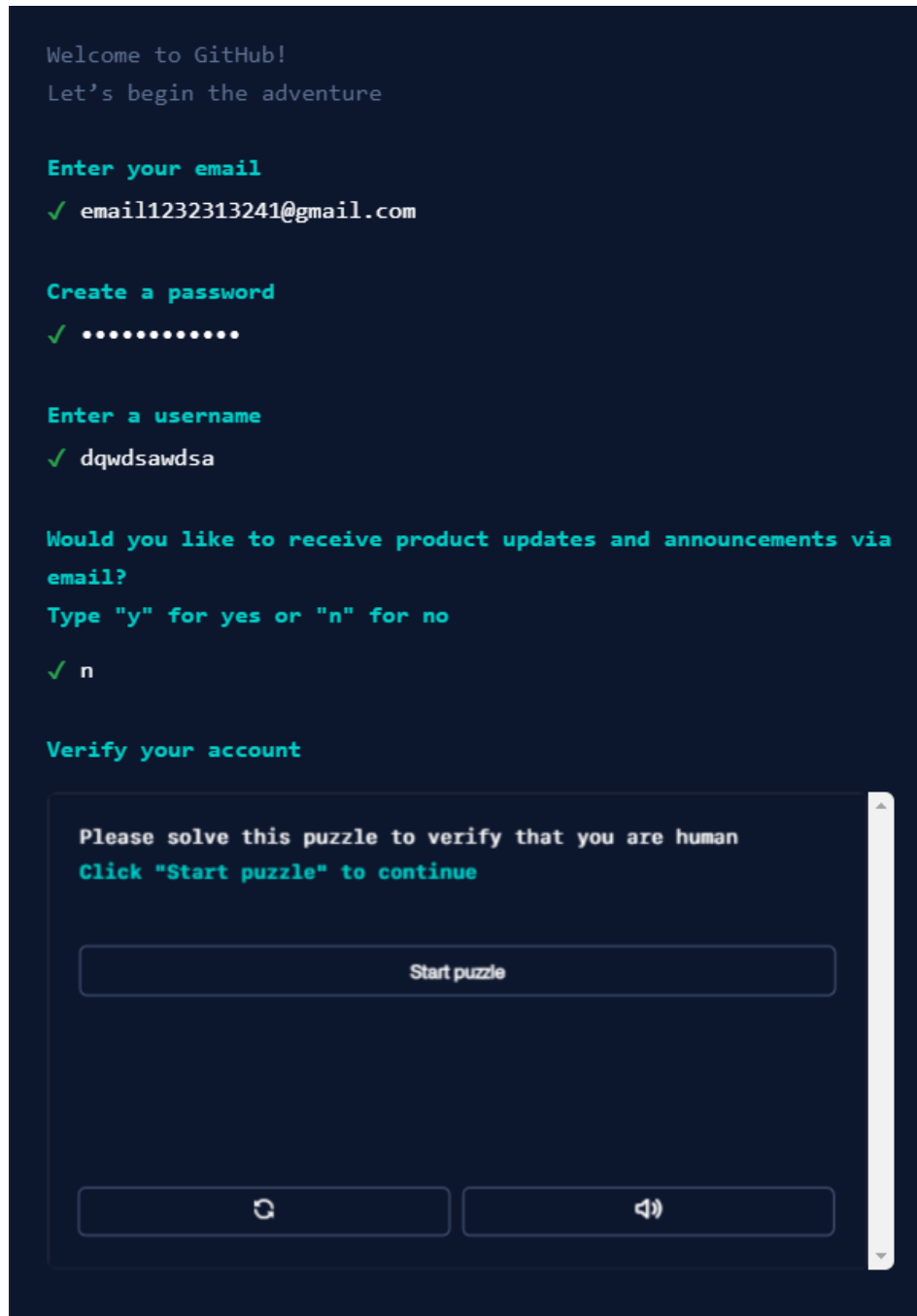


A continuación optamos por la creación de una cuenta en GitHub.

Antes de esto aquí una breve explicación de que es GitHub.
GitHub es una forja para alojar proyectos utilizando el sistema de control de versiones Git. Se utiliza principalmente para la creación de código fuente de programas de ordenador.



La creación de una cuenta en GitHub es un proceso sencillo y rápido en el cual pondremos nuestro email, una contraseña y un usuario que podrá ser visto por el resto de usuarios. Y una posterior verificación de la cuenta a través de nuestro email.



Welcome to GitHub!
Let's begin the adventure

Enter your email
✓ email1232313241@gmail.com

Create a password
✓

Enter a username
✓ dqwdsawdsa

Would you like to receive product updates and announcements via email?
Type "y" for yes or "n" for no
✓ n

Verify your account


Please solve this puzzle to verify that you are human
Click "Start puzzle" to continue

Start puzzle

⌂ 🔊


Nuevamente continuaremos con la creación de un repositorio que es un espacio donde se almacena, organiza, mantiene y difunde información digital.


Owner * Repository name *

 pazossardellagonzalo ▾ /

Great repository names are short and memorable. Need inspiration? How about [redesigned-spoon?](#)

Description (optional)

☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

☐ **Add a README file**
This is where you can write a long description for your project. [Learn more.](#)

☐ **Add .gitignore**
Choose which files not to track from a list of templates. [Learn more.](#)



☐ **Choose a license**
A license tells others what they can and can't do with your code. [Learn more.](#)

Aquí pondremos el nombre para nuestro repositorio además si queremos publicarlo de forma privada o para todo el mundo.

Una vez creado nuestro repositorio nos disponemos a través de los comandos Git, exportar los archivos que queramos subir.

Comandos

Comenzaremos usando **Get Bash Here** sobre la carpeta o archivo que queramos usar para su posterior subida al repositorio en GitHub. Una vez hecho este proceso veremos una carpeta oculta llamada `.git`, esto nos muestra que el proceso se ha realizado correctamente.

 <code>.git</code>	9/22/2021 5:58 PM	File folder	
 <code>01</code>	9/22/2021 5:57 PM	Text Document	0 KB

Se nos abrirá una consola, donde pondremos el comando **git init**, este comando crea la carpeta `.git` al igual que nos crea la rama principal en la cual podremos trabajar.

```
$ git init
```

A continuación procederemos a usar el comando **git add .** o **git add <nombre del archivo>**, la diferencia entre ambos es que con el punto al final del `git add` añadimos todos los archivos dentro de una carpeta y con el otro agregamos un archivo específico.

Una vez usado este comando, podemos utilizar **git status**, este nos permite ver el estado de los archivos tanto de los añadidos como los del **commit** que explicaré posteriormente.

```
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   01.txt
```

Con el uso del comando **git commit -m "(comentario)"** este tiene como uso el confirmar como permanentes las modificaciones realizadas en una transacción, agregando a dicha confirmación un comentario con lo que creamos conveniente. Para que tanto los usuarios que lo vean como nosotros mismos sepamos o nos podamos orientar a saber qué es lo que hemos subido.

```
$ git commit -m "Primera subida"
[master (root-commit) b8e14da] Primera subida
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 01.txt
```

Para poder subir el archivo o archivos a nuestro repositorio debemos vincular tanto nuestra cuenta de usuario de GitHub como con la URL del repositorio a nuestro ordenador.

```
$ git remote add origin https://github.com/pazossardellagonzalo/Git.git
```

Una vez ejecutado se nos abrirá una ventana en la que se nos pedirá iniciar con nuestra cuenta.



Si alguna vez queremos cambiar la cuenta podemos usar los comandos **git config --global user.email <email>** y **git config --global user.name <usuario>**.

Ahora que tenemos vinculada la cuenta podemos dar uso al comando **git push -u origin <rama>**, esto sirve para enviar los archivos de los cuales hayamos hecho el **commit** a nuestro repositorio.

```
$ git push -u origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 227 bytes | 227.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/pazossardellagonzalo/Git.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

Como vemos el archivo se ha subido correctamente, y además nos presenta información de tanto como que persona lo ha subido al igual que hace cuanto se ha producido esta acción.

El comando **git pull** nos permite extraer y descargar contenido de un repositorio.

```
$ git pull origin master
From https://github.com/pazossardellagonzalo/Git
 * branch                master      -> FETCH_HEAD
Already up to date.
```

Si necesitamos copiar los archivos que tenemos en un repositorio ya sea porque queramos una copia o para subirlo a otro repositorio, el comando **git clone** nos puede ser bastante útil.

```
$ git clone https://github.com/pazossardellagonzalo/Git.git
Cloning into 'Git'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

Como vemos usamos **git clone <url del repositorio>** y nos permite “clonar” dicho repositorio.

Branch

```
$ git checkout -b segunda
Switched to a new branch 'segunda'
```

Con el comando **git checkout -b <nombre de la rama>**, podemos crear una rama nueva.

Automáticamente al crear la rama nos posiciona en dicha rama.

```
(segunda)
```

Si queremos cambiarnos a otra rama usaremos el comando **git checkout <nombre de la rama>**.

```
$ git checkout master
Switched to branch 'master'
D      01.txt
Your branch is up to date with 'origin/master'.
```

Para ver todas las ramas existentes se usará el comando **git branch**.

```
$ git branch
* master
```

Si por ejemplo queremos combinar una rama en mi caso la secundaria con la master podremos usar el comando **git merge master**, esto si nos encontramos en la rama secundaria.

```
$ git merge master|
```

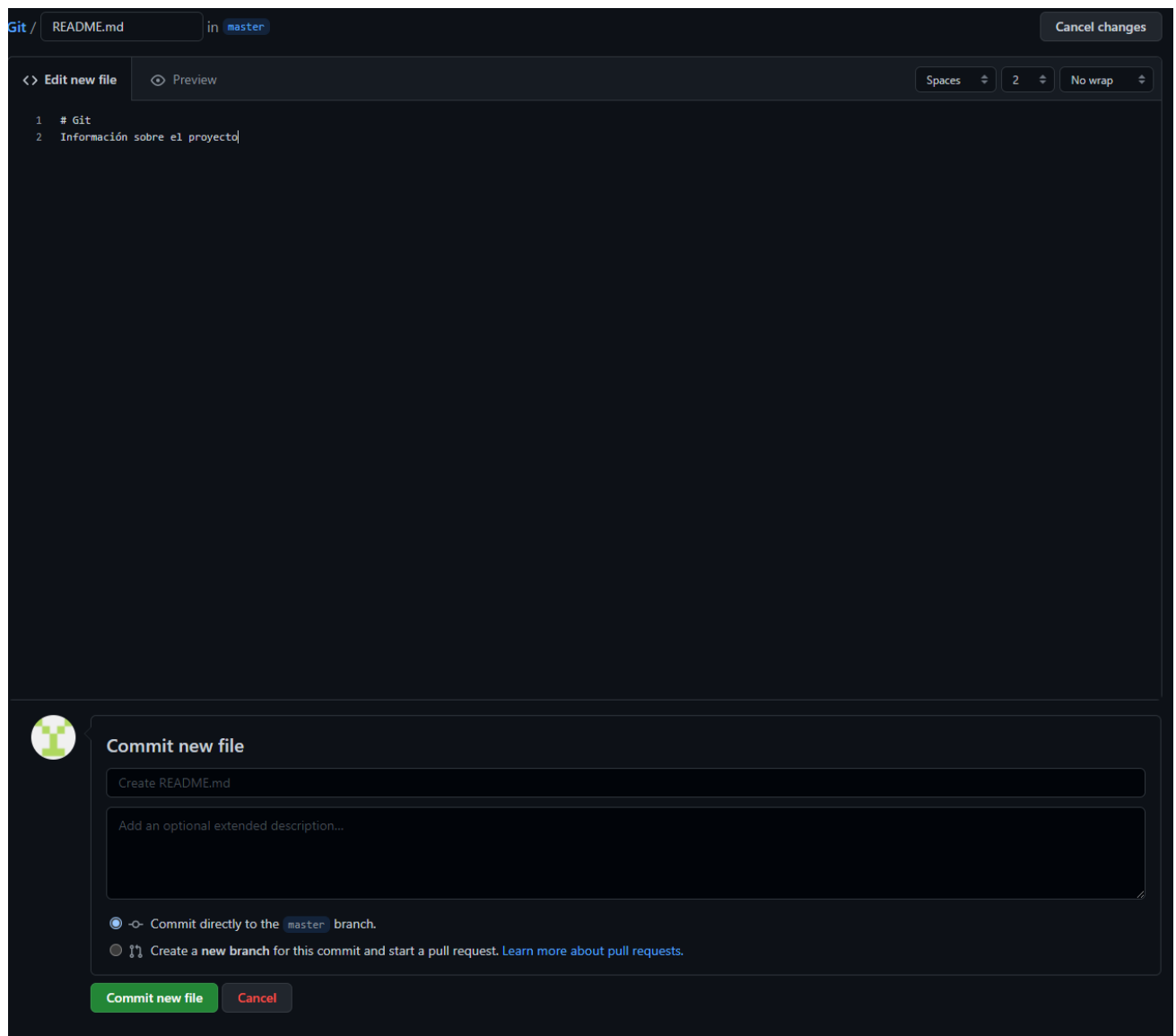
Para eliminar una de las ramas usaremos el comando **git branch -b <nombre de la rama>**.

```
$ git branch -d segunda
Deleted branch segunda (was b8e14da)
```


README

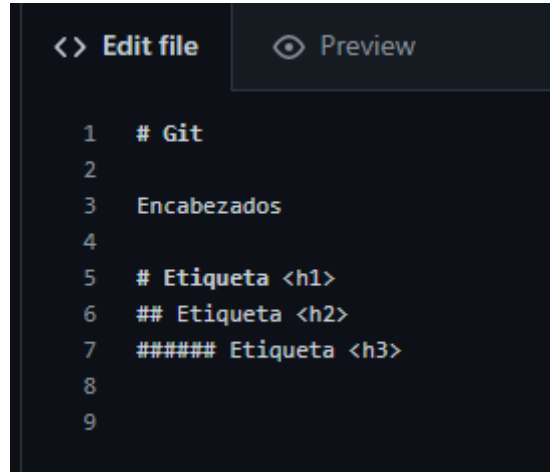
El archivo README, contiene una licencia de repositorio, pautas de contribución y un código de conducta, que nos ayudará a comunicar las expectativas y administrar las contribuciones para nuestro proyecto.

En este podemos incluir cosas tales como que hace el proyecto, porque es útil, comp pueden comenzar los usuarios el proyecto, quien mantiene y contribuye con el proyecto etc...



Markdown

Markdown es un lenguaje de forma ligera, el cual tiene una sintaxis fácil de usar, el cual podemos usar en la plataforma GitHub.



```
<> Edit file Preview
1 # Git
2
3 Encabezados
4
5 # Etiqueta <h1>
6 ## Etiqueta <h2>
7 ##### Etiqueta <h3>
8
9
```

Como vemos podemos agregar tanto texto como queramos y subir el archivo automáticamente a nuestro repositorio sin necesidad de ningún comando.

Este creará una pequeña ventana en la cual se podrá ver lo escrito.



Git Ignore

Es una cualidad de git que puede ser interesante usar en algunos casos. Digamos que estamos con un bloc de nota dentro de una carpeta que queremos exportar a un repositorio git. Usando git ignore podemos hacer que al subir todos los archivos con por ejemplo **git add** . se suban todos los que hay en la carpeta menos los que hemos ignorado.

Esto lo podemos hacer llamando al archivo <.gitignore>. Automáticamente git lo reconocerá y cumplirá la función de “ignorarlo”.

 .gitignore	9/23/2021 6:10 PM	Text Document	0 KB
 001.txt	9/23/2021 6:09 PM	Text Document	1 KB

Link GitHub: <https://github.com/pazossardellagonzalo/Git>