

Aprendiendo ha usar Git y GitHub fácilmente 🚀.



Te voy ha explicar qué es Git, y después te enseñaré ha usar esta herramineta con GitHub.

¿Qué es Git? Teoría.

Git es un **Sistema de Control de Versiones**, en otras palabras, un *programa* que **administra las distintas versiones de un Proyecto**.

Cuando usas Git lo haces para controlar un proyecto que va ha crecer (en cuestiones de contenido) a medida del tiempo cambiarás archivos.

Para controlar estos cambios, usas (al menos deberías) sistemas de control de versiones, y ahí tenemos a Git.

Git se puede usar desde proyectos de código, libros, programas, etc. Puedes usar Git en cualquier proyecto que con el tiempo cambie.

También de controlar las distintas *versiones* de tu programa, podrás compartir los cambios de tu proyecto con las distintas personas que trabajan en un proyecto y ellos contigo.

Piensa, tu trabajarás con un **Repositorio** en el que están involucrados múltiples personas y podréis estar viendo, compartiendo y deshaciendo cambios entre versiones de un proyecto. ¡Viva el trabajo cooperativo!

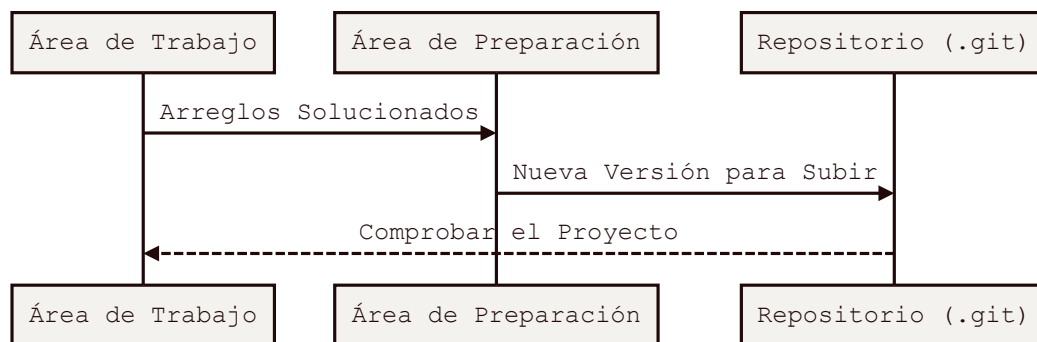
La historia de Git.

Creada por Linus Torvalds (*padre* del Kernel de Linux) en Abril de 2005. Git posee licencia GNU GPL v2, es decir... ¡Código Abierto!

Puedes saber mucho más sobre Git en [Wikipedia](#). No me extiendo más porque tampoco tiene una historia interesante, hay que ser sinceros.

El Sistema que usa Git.

Vamos a separar a Git mediante *fases/etapas*, las **Fases de Git** son las siguientes:



Puedes observar que hay 3 *estados o fases*. El primero, **Área de Trabajo** no podría tener mejor nombre; aquí es donde programas y editas. Cuando terminas de editar un archivo, este pasa al segundo estado, **Área de preparación**; este es el sitio donde están los archivos editados y listos para subir a la nueva versión de tu proyecto.

Cuando todo esté preparado, toca **subirlo al Repositorio**, la fase 3, simplemente subes los archivos que estaban en la fase 2 para que se *hagan oficiales*.

Git se puede usar tanto en forma local (en tu dispositivo) como en un servidor, ya sea propio o usando plataformas como GitHub o GitLab que son simplemente sitios donde se guarda tú código en la *nube*.

Cómo puedo yo también usar Git

Lo puedes hacer de una forma muy sencilla, vete a [este enlace](#), descargas el instalador y lo instalas. Después (si todo ha ido bien, es fácil) tendrás en tu listado de apps una llamada **Git Bash**, esto en Windows (en MacOS y Linux es simplemente la terminal).

Si abres Git Bash o la terminal (según), no creas que es un virus, hay que trabajar de esta forma (al menos por ahora). Luego veremos cómo utilizar [GitHub Desktop](#).

Puedes ver aún más información sobre cómo descargar Git (si has tenido algún problema, no funciona, etc.) visita [esta página](#).

¿Cómo puedo saber si está todo correcto?

Es realmente sencillo, en la consola o en Git Bash simplemente tecleas:

```
git --version
```

O también puedes usar en Linux, MacOS, UNIX... este:

```
which git
```

Si acabas de hacer lo que he dicho lo tendrás muy fácil desde ahora.

Vamos a crear un proyecto y si quieres cógete uno propio pero lo que necesitas es tener/crear un proyecto.

Antes de que empecemos vamos a decirle a Git algo de nosotros, como el nombre y el correo electrónico y lo hacemos de la siguiente manera:

```
git config --global user.name "Tu nombre (Ronaldo)"  
git config --global user.email ronaldo@ejemplo.com
```

Empezando con Git, paso a paso.

Los *comandos* básicos de Git son los siguientes:

- `git init` Indica a Git que va a ser usado en dicho proyecto.
- `git add <nombre_del_archivo>` *Mueve* un archivo al Área de Preparación (etapa 2) desde el Área de Trabajo (etapa 1).
- `git status` Muestra en cual etapa están tus archivos.
- `git commit` *Mueve* todos los archivos del Área de Preparación (etapa 2) a la etapa 3, la nueva versión del proyecto, el Repositorio que guarda todo el proyecto en un archivo **.git**
- `git push` Envía el archivo `.git` a un servidor remoto, ya sea tuyo o alguna plataforma como GitHub.
- `git pull` Descarga el archivo `.git` desde un servidor para que los puedas editar. También trae todo el trabajo de tus compañeros de proyecto.
- `git clone` Descarga un repositorio de un servidor para que lo puedas editar como un *fork*, un proyecto de alguien que lo editas.

La terminal (CLI) de Git posee diferentes funcionalidades o comandos, hay más comandos (si desdesas verlos puedes escribir `git --help`).

Una ayuda, si estás usando algún comando y quieres saber más sobre este, lo puedes hacer así:

```
git COMANDO --help
```

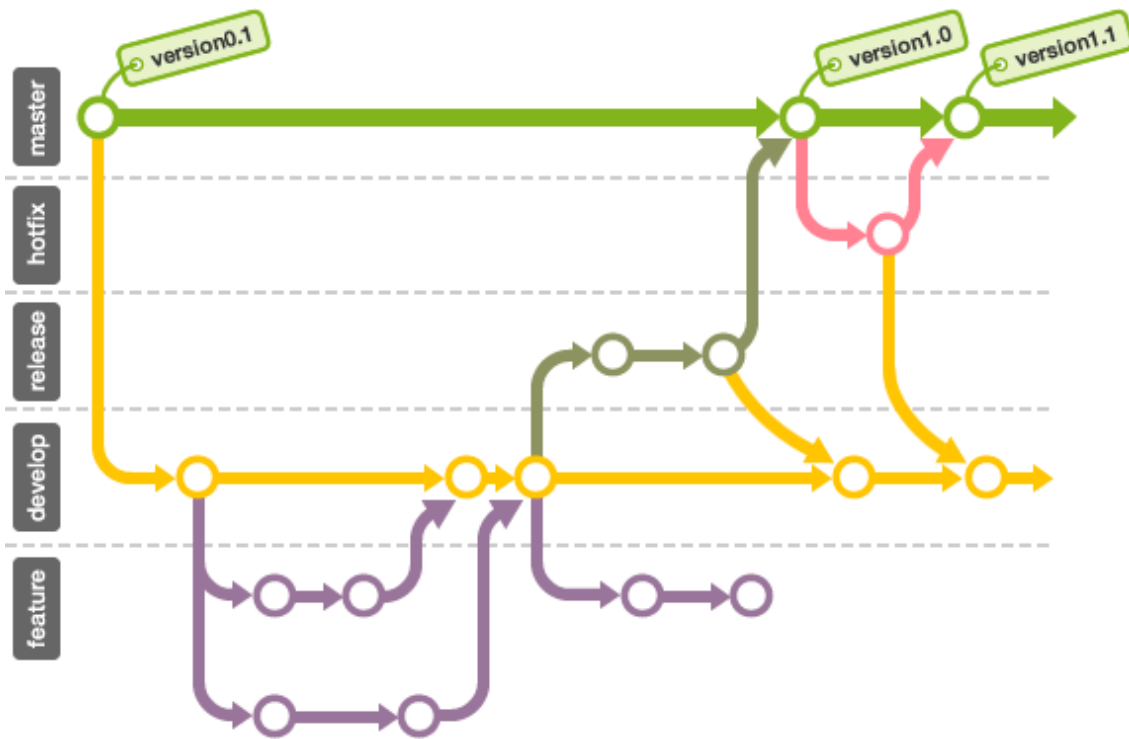
Git Branch

Veámos este concepto con un ejemplo:

Imagínate que tu proyecto se ramifica en dos partes, ej.: tus programadores crean una nueva versión de la app pero otro crea *algo* que es bueno, pues ahí es dónde se crearía un branch.

Es un sistema de *ramas* con el que puedes trabajar con más de una posibilidad a la vez. El branch por defecto se llama *master*.

Aquí puedes ver una imagen con un ejemplo visual de como funcionan los branches:



Ejemplo visual de Git Branches.

Puedes ver la rama verde es la rama principal (*master*) y cuando se está desarrollando una nueva funcionalidad o lo que sea se mueve a otro branch para no afectar a la app y, cuando esta funcionalidad no tiene errores se alade a la rama *master*.

Puedes enterarte de más sobre los branches aquí: <https://git-scm.com/book/es/v1/Ramificaciones-en-Git-%C2%BFQu%C3%A9-es-una-rama%3F>.

Cómo crear, clonar, editar y subir a GitHub mi primer proyecto.

En esta parte del libro vas ha aprender muchos comandos útiles de Git.

Crear Proyecto

Crear un proyecto es muy sencillo, puedes empezar desde cero o usar uno existente, lo único que hay que hacer para indicar a Git que tu proyecto va ha usar el control de versiones de Git es ir a la carpeta de tu proyecto, abrir la consola de Git (o la terminal), situándose en la carpeta de tu proyecto y escribe el siguiente comando:

```
git init
```

Cuando termine de ejecutarse el comando, tiene que ir muy rápido, verás que se han añadido unos archivos a tu proyecto. ¡Dejalos! Git los necesitas.

Esto también crea una carpeta oculta llamada *.git* que contiene *cosas* necesarias para que Git funcione.

Por ejemplo, para enseñarte Git tengo un proyecto, es un poema, este (por ahora) tiene un archivo llamado *poema.md* y dentro he escrito:

```
# Mi poema
```

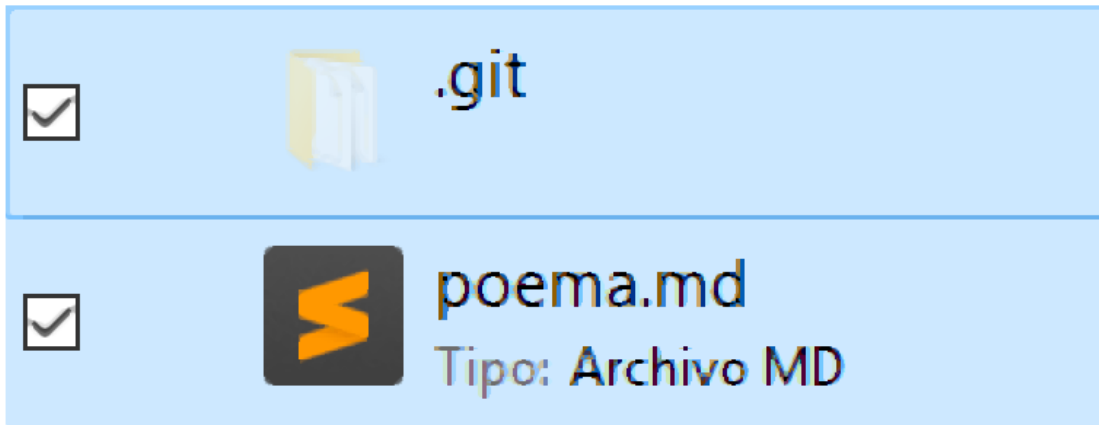
```
Este es un poema,  
que no te engañe tu mamá.
```

```
Te puede parecer muy muy estúpido  
pero es que tengo muchas cosas to do.
```

```
|
```

Para empezar me parece muy buen *experimento*, así pues seguiré así con este proyecto editándolo para tí para que aprendas la ayuda que nos puede dar Git.

Una vez ejecutado este comando en nuestra carpeta nos quedará esto:



Clonar Proyecto

Clonar es simplemente hacer una copia de un proyecto y descargarlo en tu dispositivo para editarlo (o hacer lo que quieras).

Para poder copiar un proyecto es muy sencillo, la sintaxis es la siguiente:

```
git clone <url_del_repositorio.git>
```

Así lo que estarías haciendo, es, en la carpeta en la que estás (ej.: *Escritorio*/) lo que haces es copiar un repositorio con el mismo nombre que el del repositorio, para poner uno específico tienes que escribir:

```
git clone URL_DEL_REPOSITORIO.git NOMBRE_CARPETA
```

Puedes clonar un proyecto de dos formas, desde tu ordenador o también lo puedes descargar desde un servidor como GitHub. Ej:

```
git clone https://github.com/RonaldoDavid/git-book.git
```

Controlar archivo/s y/o carpeta/s del Proyecto

Para editar un proyecto hay que acordarse de las tres fases por las que un archivo puede estar. Si no las recordáis son: **Área de Trabajo**, **Área de Preparación** y el **Repositorio con extensión .git**, es muy fácil.

Para pasar un archivo, una carpeta o todo de la del Área de Trabajo al Área de Preparación para después crear la nueva versión (que lo verás en la siguiente sección) se hace con los siguientes comandos:

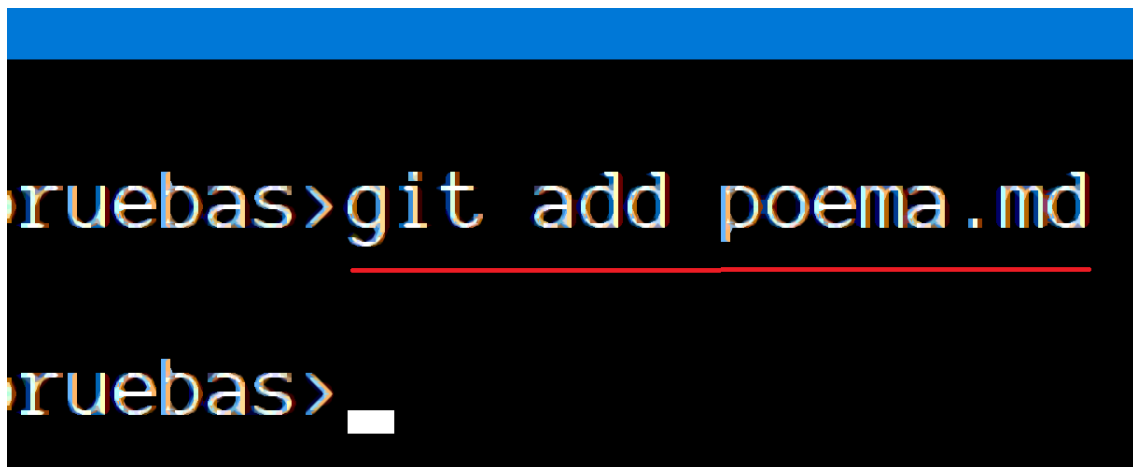
```
git add <archivo1/carpeta_1> <archivo2/carpeta_2> <...>
```

Aquí puedes añadir archivos o carpetas puedes añadir uno o más (ej.: `git add poema.md` , `git add poema.md romance.md`).

O si deseas puedes añadir todo el proyecto de esta forma:

```
git add .
```

Por ejemplo, tengo esta nueva versión de mi *poema.md*, así pues quiero añadirla, lo haría (en mi terminal) mediante:



Si todo ha ido bien no debería decirte nada, si ha habido algún problema, te dirá dónde y porqué.

Saber cómo están nuestros archivos

Con el siguiente comando podrás saber si tu proyecto está en la fase 1 (Área de Trabajo) o fase 2 (Área de Preparación).

```
git status
```


Crear un branch (rama) en nuestro Proyecto.

Es muy sencillo hacer un branch, simplemente lo haces así:

```
git branch NOMBRE_DEL_BRANCH
```

Fusionar lo que antes habíamos ramificado (merge).

Cuando una funcionalidad ya está correctamente integrada, toca pasarla a la rama principal, para hacerlo haríamos:

```
git merge <BRANCH>
```

Si tienes problemas con *git merge*.

Si estás teniendo problemas al combinar dos branches puedes ejecutar este comando que te ayudará:

```
git mergetool
```

En [este enlace](#) puedes ver los problemas típicos que hay al hacer esta fusión.

Cambiar de ramas (checkout).

Con este comando podemos *movernos* fácilmente entre ramas:

```
git checkout <NOMBRE_DEL_BRANCH_AL_QUE_VAMOS>
```

Crear la nueva versión (.git) de nuestro Proyecto.

Para hacer eso antes tienes que haber hecho cambios en el Proyecto y haberlos añadido a la fase 2 (Área de Preparación), una vez lo hayas hecho, tienes que escribir este sencillo comando:

```
git commit -m "Descripción (mejoras, arreglos, etc.)"
```

Con esto crearás la nueva versión de tu Proyecto. Este es el resultado:

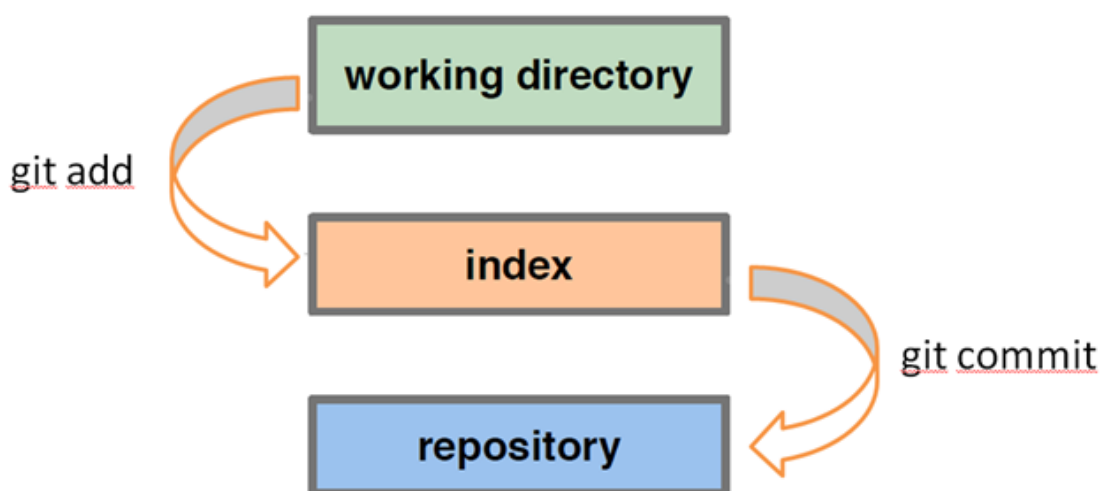
```
git commit -m "Esta es la siguiente versión de nuestro poema"
```

Y al ejecutar el comando me dice:

```
[master (root-commit) f8b031e] Esta es la siguiente  
versión de nuestro poema  
1 file changed, 11 insertions(+)  
create mode 100644 poema.md
```

Por ahora no te preocupes si no entiendes algo, simplemente me está diciendo que he creado una nueva versión de mi proyecto y que se ha editado un archivo y se han insertado 11 líneas. Solo eso, no tienes que entender nada más.

Esta es la *misma imagen* pero en Inglés del diagrama que tienes al principio del Libro.



Puedes ver *Commit*, pues eso es lo que hemos hecho. Fácil.

Ahora vamos a usar GitHub para guardar nuestros repositorios de forma gratis (sin que tu seas el producto) y en la *nube* para que nunca pierdas la información.

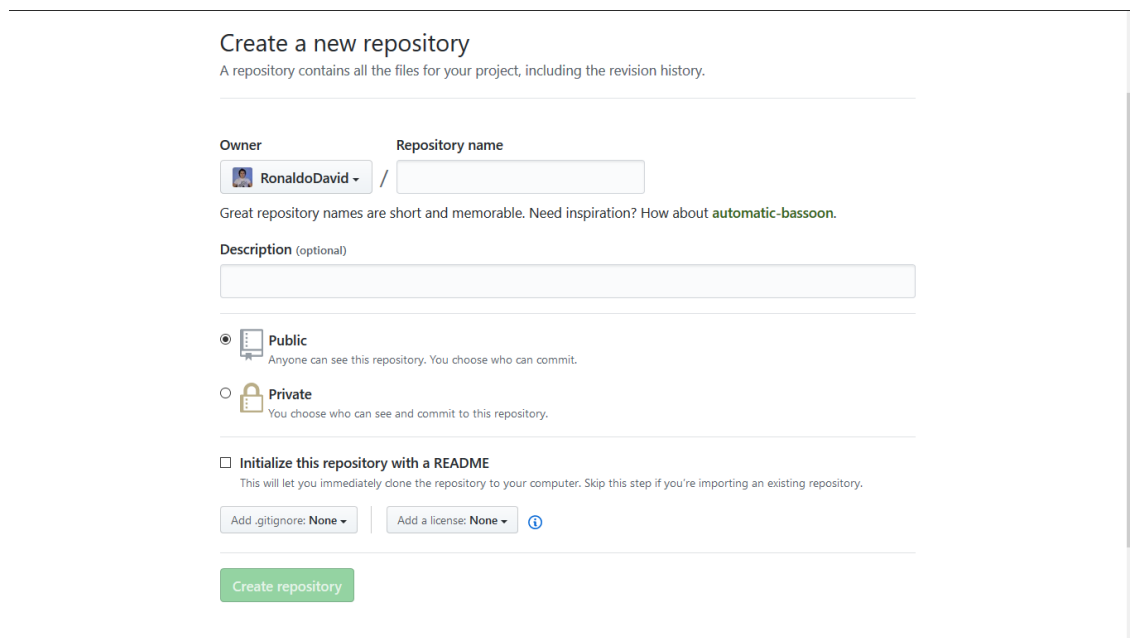
Usar GitHub para guardar nuestros Proyectos.

GitHub

GitHub es una plataforma en la que puedes alojar código de forma gratuita.

Para poder usar esta plataforma tienes que crear una cuenta, puedes hacerlo desde este enlace <https://github.com/join> (selecciona una cuenta gratuita por ahora, pero si quieres pagar... no te digo nada).

Una vez hecho todos los pasos del registro, te encontrarás un *Panel*, algo por el estilo. Crea un repositorio, para hacerlo te diriges al menú superior y presionas en el botón + > *New Repository* o te diriges a <https://github.com/new>.



The screenshot shows the 'Create a new repository' page on GitHub. At the top, it says 'Create a new repository' and 'A repository contains all the files for your project, including the revision history.' Below this, there are two input fields: 'Owner' (with a dropdown menu showing 'RonaldoDavid') and 'Repository name' (an empty text box). A note below these fields says 'Great repository names are short and memorable. Need inspiration? How about [automatic-bassoon](#).' There is a 'Description (optional)' text area. Below that, there are two radio button options: 'Public' (selected) and 'Private'. The 'Public' option has a subtext: 'Anyone can see this repository. You choose who can commit.' The 'Private' option has a subtext: 'You choose who can see and commit to this repository.' Below these, there is a checkbox labeled 'Initialize this repository with a README' and a subtext: 'This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.' At the bottom, there are two dropdown menus: 'Add .gitignore: None' and 'Add a license: None', followed by a 'Create repository' button.

Aquí escribes un nombre para el repositorio, una descripción y si quieres que sea privado tienes que pagar, *c'est la vie* y marcas la casilla ***Initialize this repository with a README***. Y por aquí ya hemos terminado.

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner

RonaldoDavid

Repository name

git-example

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

Description (optional)

Este es Repositorio de Pruebas para el libro de Git.

☒ Public

Anyone can see this repository. You choose who can commit.

☐ Private

You choose who can see and commit to this repository.

☒ Initialize this repository with a README

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None

Add a license: None

Create repository

Esta es la interfaz de un repositorio en GitHub:

Search or jump to...

Pull requests Issues Marketplace Explore

Watch 0 Star 0 Fork 0

RonaldoDavid / git-example

<> Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

Este es Repositorio de Pruebas para el libro de Git. Edit

Manage topics

1 commit 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download

RonaldoDavid Initial commit Latest commit b85d93b just now

README.md Initial commit just now

README.md

git-example

Este es Repositorio de Pruebas para el libro de Git.

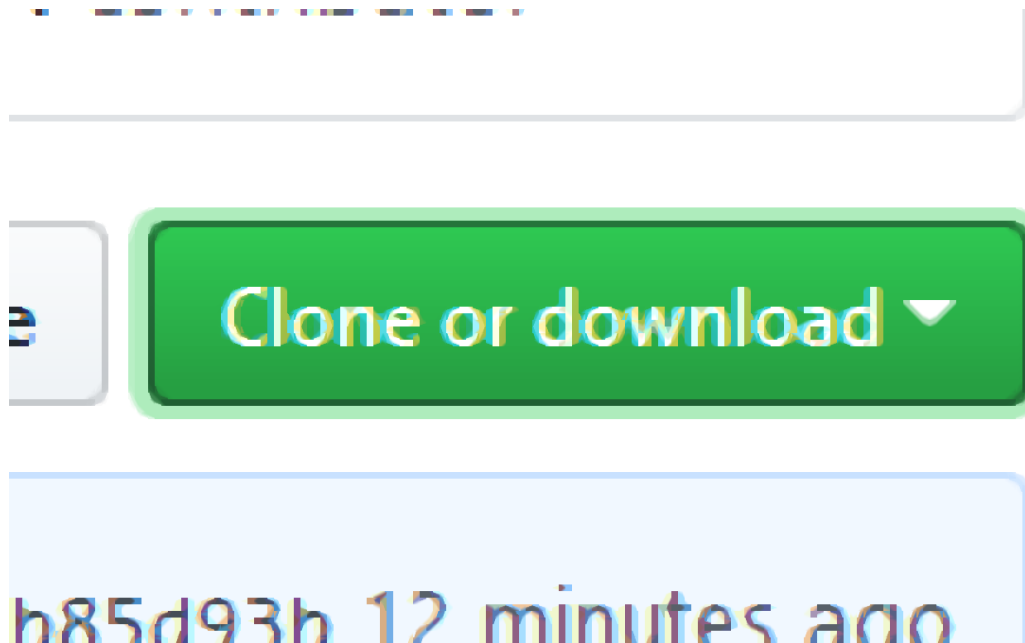
Usar Git con la mano de GitHub.

Para poder usar GitHub con Git tienes que tener abierta la página del repositorio en GitHub y también tener abierta la terminal en la carpeta del proyecto.

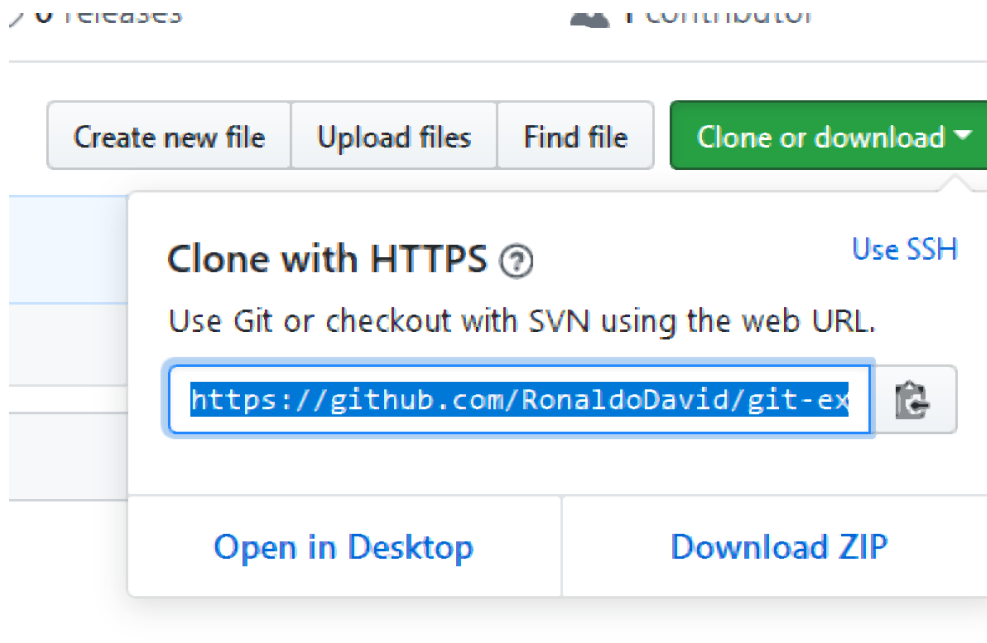
Aquí escribirás los siguientes comandos:

```
git remote add origin URL_DEL_RESPOSITORIO
```

La `URL_DEL_REPOSITORIO` la puedes encontrar en la página del repositorio en un botón verde llamado ***Clone or download***:



Lo clicas y se desplegará un menú como el siguiente:



Y tendrás que copiar el enlace al sitio dónde está el `.git` de tu repositorio.

Una vez escrito el primer comando (en mi caso `git remote add origin https://github.com/RonaldoDavid/git-example.git`) toca escribir esto:

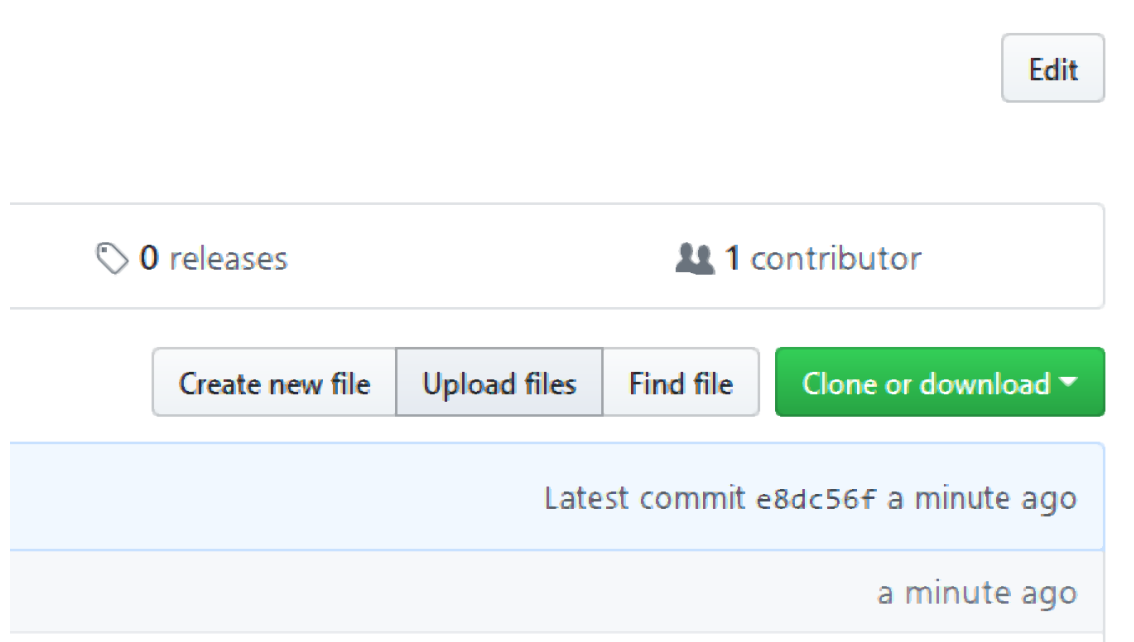
```
git remote -v
```

Y para terminar lo *subes* con este último comando:

```
git push origin master
```

Aparecerá una pequeña ventana en la que tendrás que escribir tu usuario/email y contraseña de GitHub y se tendrá que subir, si tienes algún error es posible que sea por el archivo *README.md* simplemente creas uno en tu proyecto, haces el `git add README.md` y el `git commit -m "Añadiendo el README"` y ya está.

Si no te quieres complicar la vida o yo que se... te apetece, lo puedes hacer desde GitHub y santas pascuas. Observa la imagen:



Lo haces simplemente desde el grupo de botones que tienes arriba del listado de archivos > **Upload Files**

GitHub Desktop.

GitHub desktop es una aplicación para Windows, MacOS y Linux en la que puedes hacer (casi) todo lo que haces en la terminal y en la web en una simple app, desarrollada por GitHub. Aquí tienes una captura de la app:

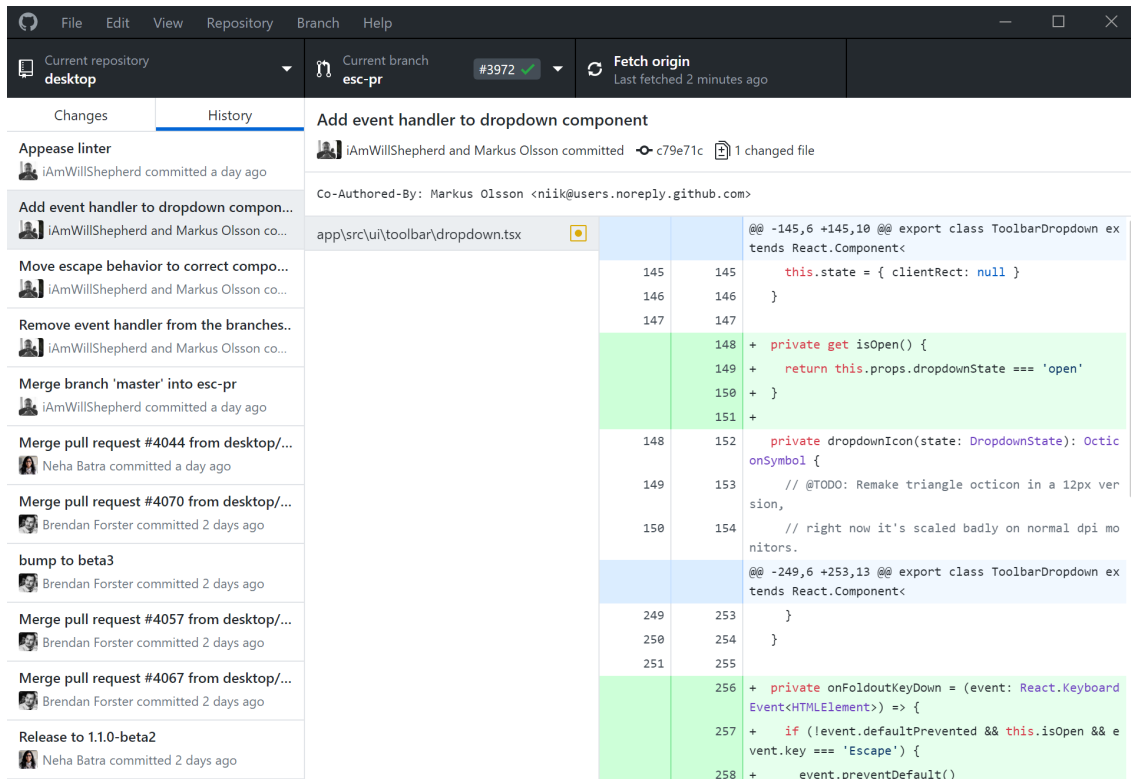


Foto de <https://desktop.github.com/>

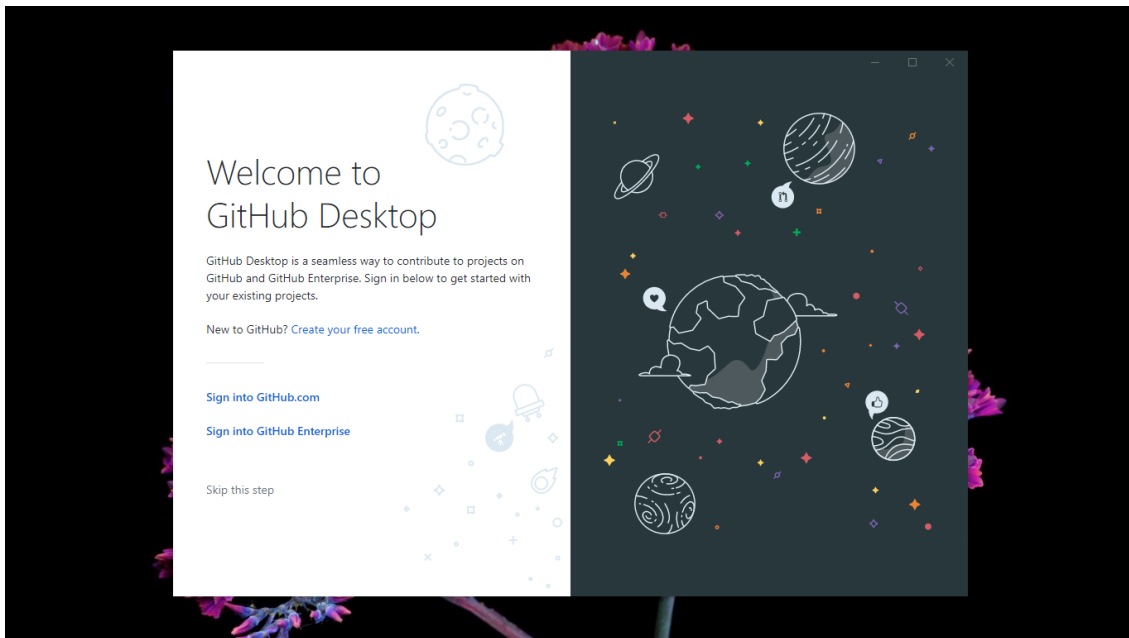
Vamos a infagar un poco más en esta potente app ahora mismo.

Esta app posee muchas funcionalidades, como por ejemplo la posibilidad de reemplazar totalmente a la terminal de Git y ver de una forma mucho más visual que Git (y quizá GitHub).

Cada vez que edites un archivo de tu proyecto te saldrá en un menú la opción de seleccionarlo para hacer el *Commit*. De una forma muy sencilla lo podrás hacer, ya que es casi todo automático.

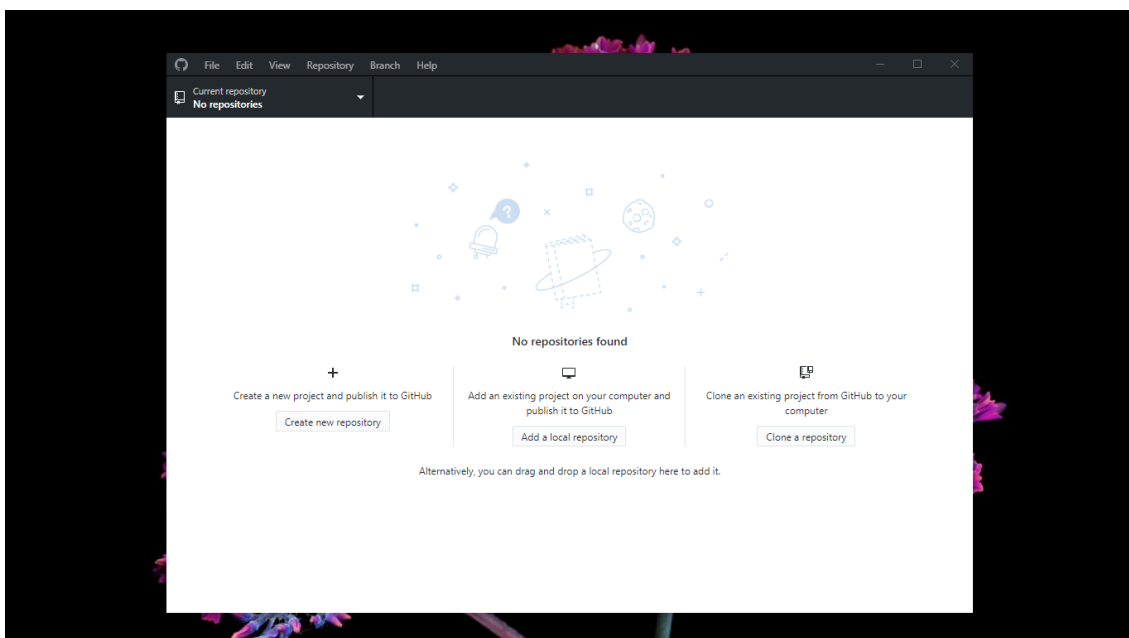
Es muy (incluso demasiado) fácil de usar, ya que posee una interfaz muy clara y aunque no esté en español es muy fácil de entender.

Cuando la instalación termine tendrás la siguiente pantalla para entrar con tu cuenta de GitHub:



Interfaz de Inicio de Sesión de GitHub Desktop.

Una vez que inicies sesión, te aparecerá la siguiente pantalla en la que tendrás tres opciones:



Nosotros vamos a seleccionar la segunda opción (la del centro de la pantalla) que es *vincular* el repositorio con la carpeta local. Aquí la puedes ver mejor:

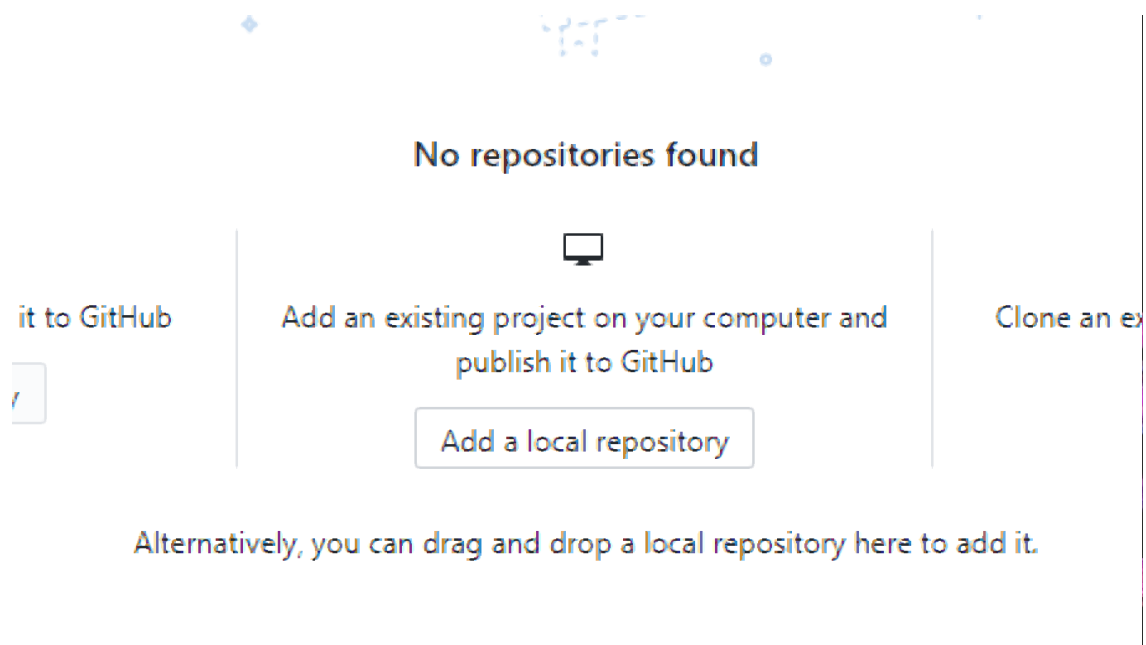
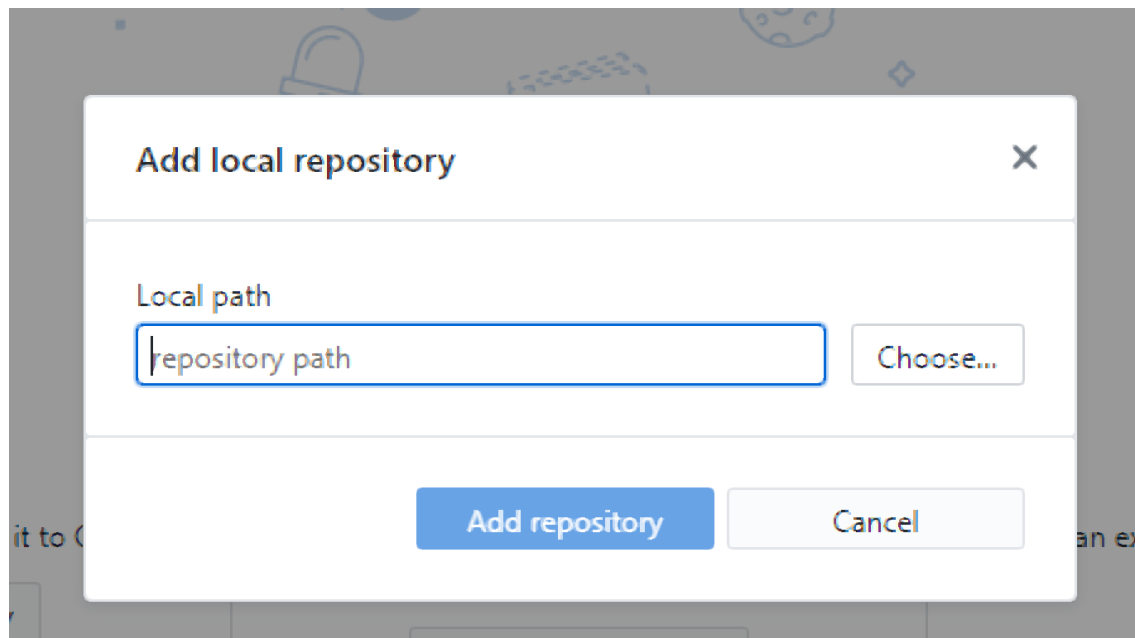


Imagen con zoom de la segunda opción.

Cuando hayas clicado en el botón de la segunda opción te saldrá una mini ventana en la que tendrás que seleccionar el directorio de tu proyecto con Git para vincularlo al repositorio de GitHub:



Selector de archivos de GitHub, para vincular el repo. con la carpeta local mediante el botón *Choose*

Cuando eso esté listo (es tarea super sencilla) lo tendrás todo preparado para remplazar la terminal por tu nuevo sistema de control de versiones *algo diferente* pero te acostumbras de forma fácil y rápida.

Esta herramienta no es la única, muchos editores de código como Visual Studio Code o Atom (ambos gratuitos) que poseen una herramienta de Git y GitHub integrada sin necesidad de instalar extensiones o complementos.

Historia de GitHub.

GitHub nace en el año 2008 a manos de Tom Preston-Werner, Chris Wanstrath y PJ Hyett. La intención que tenían era crear una plataforma de desarrollo colaborativo usando el sistema de Git y lo consiguieron.

El 4 de Junio de 2018, Microsoft compró GitHub por la cantidad de 7.500 millones de dólares. A GitHub hay alternativas como [GitLab](#).

Written by Ronaldo David Gomes Gligan, [gligan.info](#).

License **CC BY-SA 4.0**.