

CURSO DE PROGRAMACIÓN FULL STACK

# TUTORIAL: GUÍA PRÁCTICA DE USO DE GIT CON GITHUB



**EGG**

# GUÍA PRÁCTICA DE USO DE GIT CON GITHUB

## ¿QUÉ ES EL CONTROL DE VERSIONES?

Los **sistemas de control de versiones** son una categoría de herramientas de software que ayudan a un equipo de software a **gestionar los cambios** en el código fuente a lo largo del tiempo. El software de control de versiones realiza un seguimiento de todas las modificaciones en el código en un tipo especial de base de datos. Si se comete un error, los desarrolladores pueden ir atrás en el tiempo y comparar las versiones anteriores del código para ayudar a resolver el error al tiempo que se minimizan las interrupciones para todos los miembros del equipo.

Los desarrolladores que trabajan en equipos están escribiendo continuamente nuevo código fuente y cambiando el que ya existe. El código de un proyecto, una aplicación o un componente de software normalmente se organiza en una estructura de carpetas o "árbol de archivos". Un desarrollador del equipo podría estar trabajando en una nueva función mientras otro desarrollador soluciona un error no relacionado cambiando código. Cada desarrollador podría hacer sus cambios en varias partes del árbol de archivos.

El control de versiones ayuda a los equipos a resolver estos tipos de problemas, al realizar un *seguimiento* de todos los cambios individuales de cada colaborador y ayudar a evitar que el trabajo concurrente entre en conflicto

En definitiva, tener un control de los cambios en los códigos de nuestra aplicación es una variable crucial para el éxito de nuestro desarrollo. **Git** es un sistema de control de versiones de código abierto, diseñado para manejar grandes y pequeños proyectos con rapidez y eficiencia. La pretensión de este tutorial es abordar el uso básico de Git proporcionando ejemplos prácticos útiles para comenzar a administrar repositorios remotos con plataformas como **Bitbucket** o **GitHub**.

## CONTROL DE VERSIONES CON GIT

Git es la mejor opción para la mayoría de los equipos de software actuales. Aunque cada equipo es diferente y debería realizar su propio análisis, aquí recogemos los principales motivos por los que destaca el control de versiones de Git con respecto a otras alternativas:

### GIT ES UNA EXCELENTE HERRAMIENTA

Git tiene la funcionalidad, el rendimiento, la seguridad y la flexibilidad que la mayoría de los equipos y desarrolladores individuales necesitan. En las comparaciones directas con gran parte de las demás alternativas, Git resulta muy ventajoso para muchos equipos.

# GIT ES UN PROYECTO DE CÓDIGO ABIERTO DE CALIDAD

Git es un proyecto de código abierto muy bien respaldado con más de una década de gestión de gran fiabilidad. Los encargados de mantener el proyecto han demostrado un criterio equilibrado y un enfoque maduro para satisfacer las necesidades a largo plazo de sus usuarios con publicaciones periódicas que mejoran la facilidad de uso y la funcionalidad. La calidad del software de código abierto resulta sencilla de analizar y un sinnúmero de empresas dependen en gran medida de esa calidad.

Git goza de una amplia base de usuarios y de un gran apoyo por parte de la comunidad. La documentación es excepcional y para nada escasa, ya que incluye libros, tutoriales y sitios web especializados, así como podcasts y tutoriales en vídeo.

El hecho de que sea de código abierto reduce el costo para los desarrolladores aficionados, puesto que pueden utilizar Git sin necesidad de pagar ninguna cuota. En lo que respecta a los proyectos de código abierto, no cabe duda de que Git es el sucesor de las anteriores generaciones de los exitosos sistemas de control de versiones de código abierto, SVN y CVS.

## INSTALACIÓN DE GIT

### INSTALADOR DE GIT PARA MAC

La forma más sencilla de instalar Git en un Mac es mediante el instalador independiente:

1. Descarga el instalador de Git para Mac más reciente desde acá:  
<https://sourceforge.net/projects/git-osx-installer/files/>
2. Sigue las instrucciones para instalar Git.
3. Abre un terminal y escribe el siguiente texto para comprobar que la instalación se ha realizado correctamente:

```
git --version:
```

```
$ git --version  
git version 2.9.2
```

Para más opciones consulta acá: <https://www.atlassian.com/es/git/tutorials/install-git#mac-os-x>

### INSTALADOR DE GIT PARA LINUX

Debian / Ubuntu (apt-get)

Los paquetes de Git están disponibles mediante APT:

1. Desde tu núcleo, instala Git mediante apt-get:

```
$ sudo apt-get update
```

```
$ sudo apt-get install git
```

2. Introduce el siguiente texto para verificar que la instalación se ha realizado correctamente:

```
git --version:
```

```
$ git --version  
git version 2.9.2
```

## INSTALADOR DE GIT PARA WINDOWS

1. Descarga el [instalador de Git para Windows](#) más reciente.
2. Cuando hayas iniciado correctamente el instalador, verás la pantalla del asistente de instalación de Git.
3. Selecciona las opciones Siguiente y Finalizar para completar la instalación. Las opciones predeterminadas son las más lógicas en la mayoría de los casos.
4. Abre el símbolo del sistema (o Git Bash si durante la instalación seleccionaste no usar Git desde el símbolo del sistema de Windows).
5. Introduce el siguiente texto para verificar que la instalación se ha realizado

```
git --version:
```

```
$ git --version  
git version 2.9.2
```

## CONFIGURACIÓN INICIAL

Abra su terminal de Git para comenzar con la ejecución de comandos, por ejemplo, abrirá el programa **Git bash** en Windows para ingresar a la línea de comandos de este programa.

Una vez que ingrese, use el siguiente comando para establecer el nombre de usuario de git:

```
git config --global user.name "Jhoel Perez"
```

Recuerde sustituir el texto entre comillas por su nombre real. Ahora indique el correo electrónico del usuario para git:

```
git config --global user.email "micorreopersonal@jhoel.com"
```

Sustituyendo el texto entre comillas por su cuenta de correo electrónico. Esta configuración inicial debería ser suficiente para comenzar. Para comprobar otros valores de su configuración actual ejecute:

```
git config --list
```

Se mostrarán los nuevos valores configurados al final, y otros valores de configuración predeterminados:

```
...
color.diff=auto
color.status=auto
...
user.name=Juan Perez
user.email=micorreopersonal@juan.com
```

## PRIMEROS PASOS

### GITHUB DESKTOP

GitHub Desktop, la aplicación de escritorio de GitHub, te permitirá empezar a utilizar un control de versiones sin problemas. GitHub Desktop es, de hecho, una Interfaz Gráfica de Usuario (GUI, por sus siglas en inglés) diseñada para facilitar el uso de Git

### INSTALAR GITHUB DESKTOP

Puedes instalar GitHub Desktop en sistemas operativos Microsoft Windows o macOS

Antes de configurar GitHub Desktop, debes contar previamente con una cuenta de GitHub o GitHub Enterprise.

- Para obtener más información sobre crear una cuenta de GitHub, consulta "Registrar una nueva cuenta de GitHub".
- Para una cuenta de GitHub Enterprise, contacta a tu administrador de sitio de GitHub Enterprise.

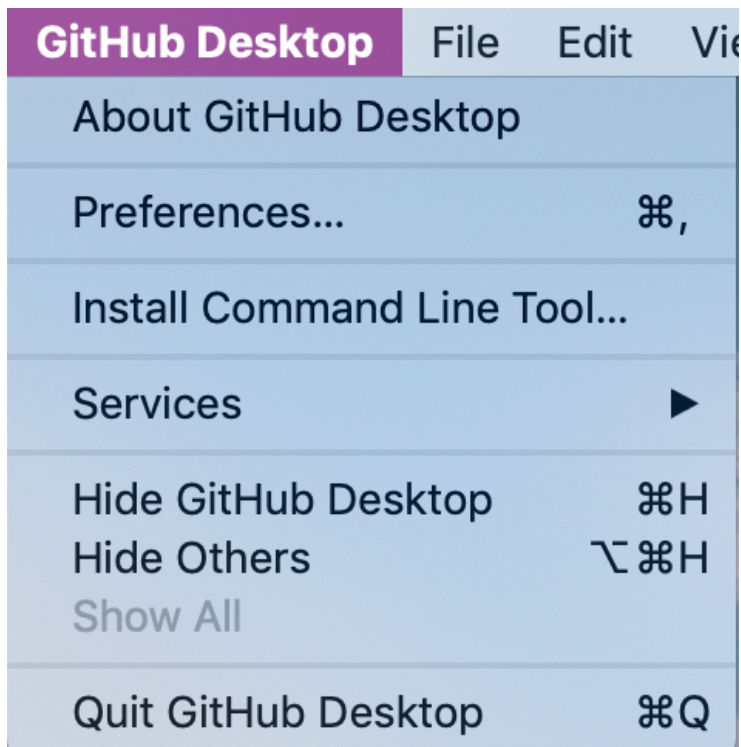
Puedes instalar GitHub Desktop en macOS 10.10 o posterior y Windows 7 o posterior.

1. Visita la página de descargas GitHub Desktop .
2. Escoge Download for (tu Sistema Operativo).
3. En la carpeta Download (Descargas) de tu computadora, haz doble clic en el archivo comprimido GitHub Desktop.
4. Una vez que se descomprima el archivo, haz doble clic en GitHub Desktop.

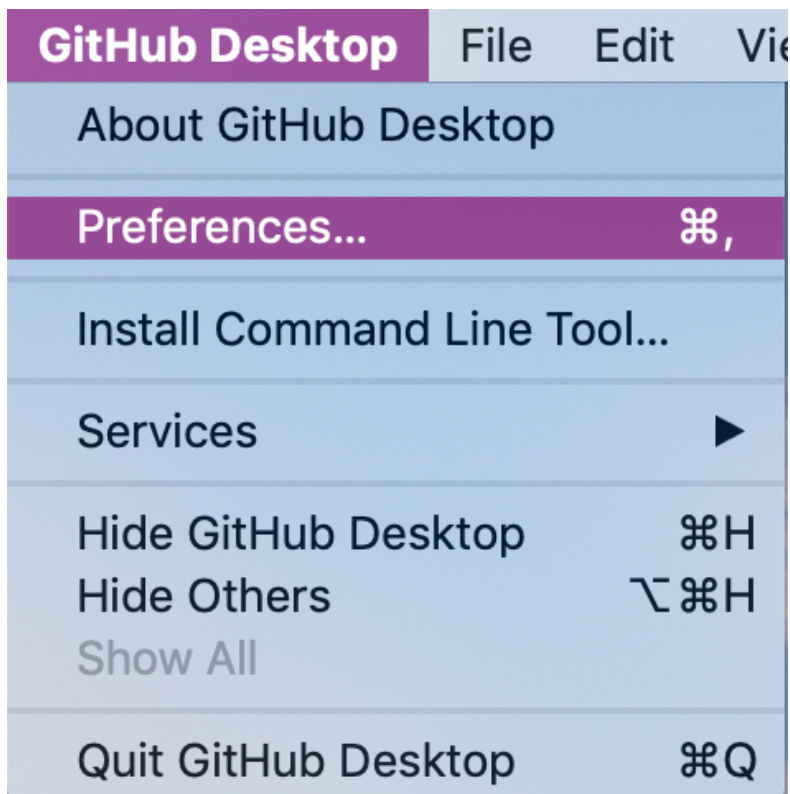
### CONFIGURACIÓN BÁSICA PARA MAC

Puedes acceder a la configuración para proteger tu privacidad, conectar cuentas a GitHub Desktop, y configurar Git.

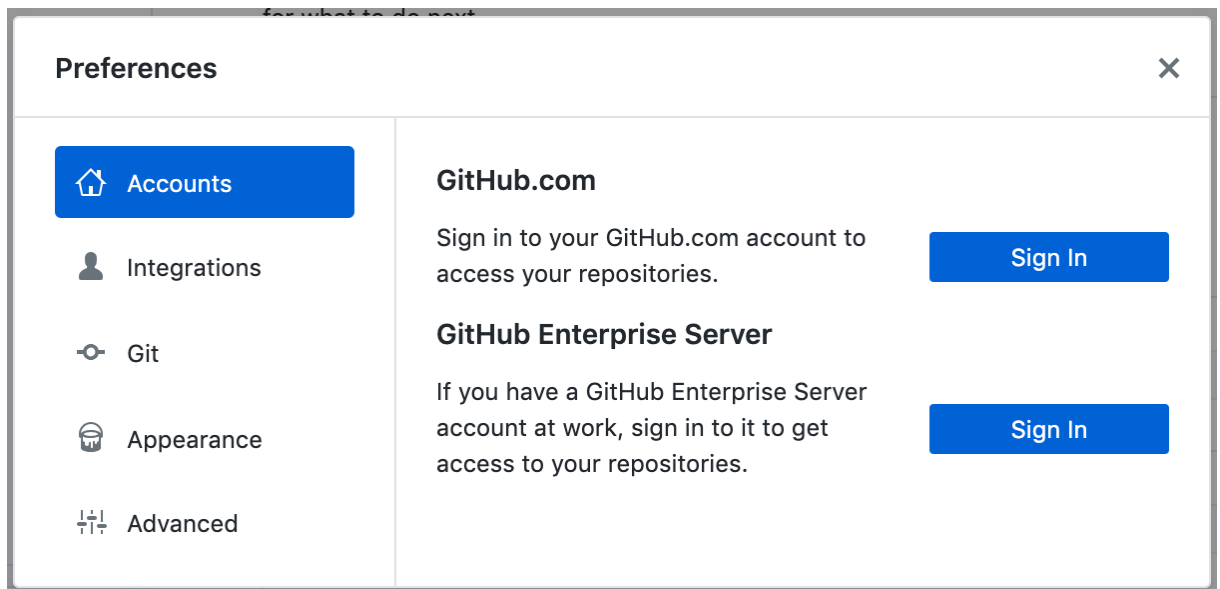
1. En la esquina superior izquierda de tu pantalla, selecciona el menú de \*\*GitHub Desktop.



2. Haz clic en Preferences(Preferencias).



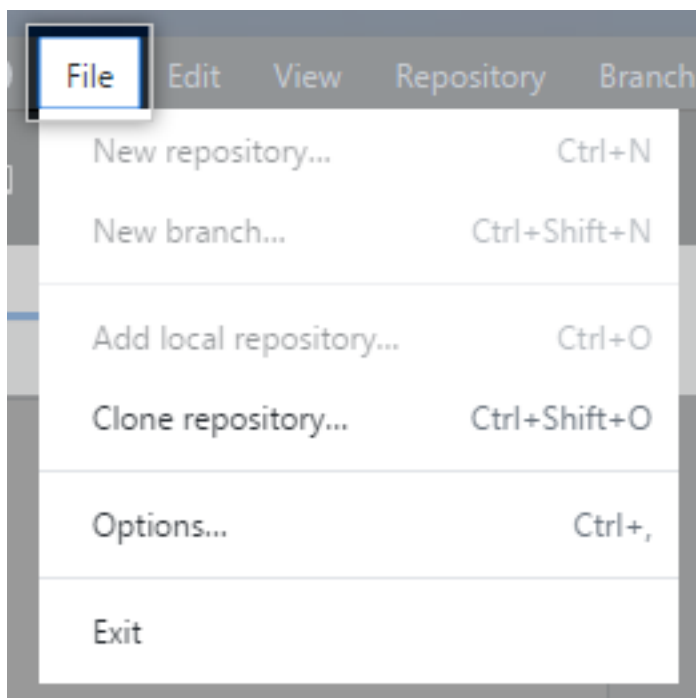
3. Para ver o cambiar tu configuración, alterna entre estos paneles:



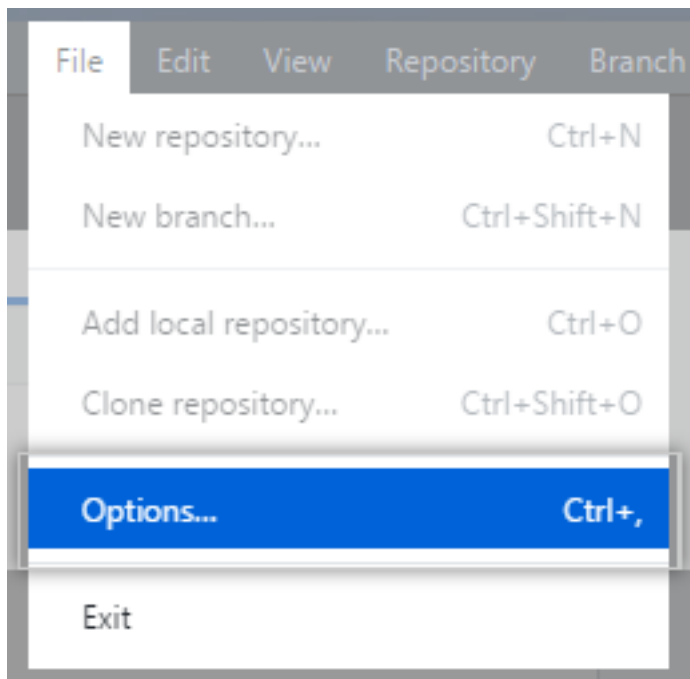
4. Elige **Accounts** (Cuentas) para agregar o quitar una cuenta GitHub o GitHub Enterprise.
5. Elija **Integrations** (Integraciones) para elegir un editor o shell.
6. Elige **Git** para editar tu configuración de Git.
7. Elige **Appearance** (Apariencia) para cambiar entre el tema claro u oscuro.
8. Elige **Advanced** (Avanzado) para más opciones de configuración.

## CONFIGURACIÓN BÁSICA PARA WINDOWS

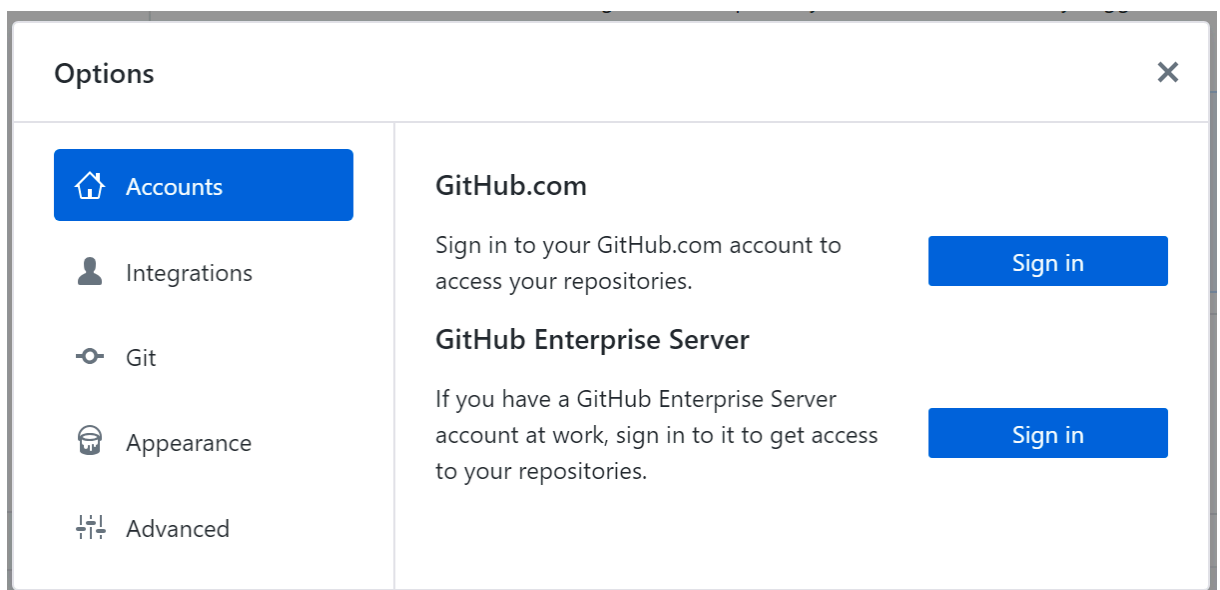
1. En la esquina superior izquierda de la ventana, selecciona el menú File (Archivo).



2. Haz clic en Options (Opciones).



3. Para ver o cambiar tu configuración, alterna entre estos paneles:



4. Elige **Accounts (Cuentas)** para agregar o quitar una cuenta GitHub o GitHub Enterprise.
5. Elige **Integrations (Integraciones)** para elegir un editor o shell.
6. Elige **Git** para editar tu configuración de Git.
7. Elige **Appearance (Apariencia)** para cambiar entre el tema claro u oscuro.
8. Elige **Advanced (Avanzado)** para más opciones de configuración.

## CREAR EL PRIMER REPOSITORIO MEDIANTE GITHUB DESKTOP

Puedes usar GitHub Desktop para comenzar a trabajar rápidamente con un repositorio Git sin necesidad de usar la línea de comando.



## INTRODUCCIÓN

Esta guía te orientará a través del proceso de uso de GitHub Desktop para trabajar en un repositorio Git. GitHub Desktop se extiende y simplifica tu flujo de trabajo GitHub.com, usando una interfaz visual en lugar de comandos de texto en la línea de comandos. Al final de esta guía, habrás usado GitHub Desktop para crear un repositorio, realizar cambios al repositorio, y publicar los cambios en GitHub.com o GitHub Enterprise Server.

Después de descargar GitHub Desktop e iniciar sesión en GitHub o GitHub Enterprise puedes crear y clonar un repositorio de tutorial. El tutorial introducirá los aspectos básicos del trabajo con Git y GitHub, que incluye la instalación de un editor, la creación de una rama, la generación de una confirmación, la extracción de GitHub.com, y la creación de una solicitud de extracción. El tutorial está disponible mientras no tengas ningún repositorio en GitHub Desktop.

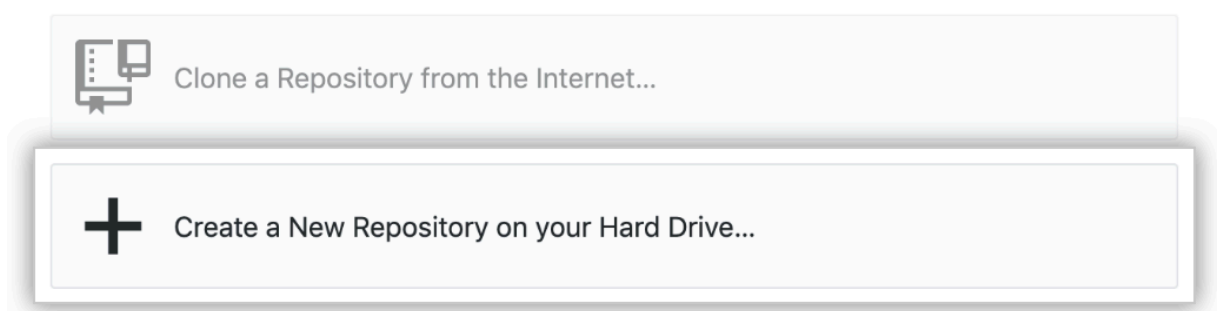
## PASO 1. INSTALAR E INICIAR SESIÓN EN GITHUB DESKTOP

1. Si aún no lo has hecho, Descarga GitHub Desktop [de https://desktop.github.com/](https://desktop.github.com/). GitHub Desktop soporta las versiones recientes de Windows y macOS. Para conocer las instrucciones de instalación específicas para tu sistema operativo, consulta [Instalar GitHub Desktop](#).
2. Abre GitHub Desktop y sigue el flujo de bienvenida inicial para iniciar sesión en tu cuenta de GitHub. Verás un paso "Configure Git" (Configurar Git), donde puedes configurar tu nombre y dirección de correo electrónico. Para asegurarte de que tus confirmaciones se atribuyan correctamente a tu cuenta de GitHub, usa la dirección de correo electrónico asociada a tu cuenta de GitHub. Para obtener más información, [consulta "Establecer tu dirección de correo electrónico de confirmación."](#)

## PASO 2. CREAR UN REPOSITORIO NUEVO

Verás una vista "Let's get started!" (¡Comencemos!)", donde puedes elegir si deseas crear y clonar un repositorio de tutorial, crear un repositorio nuevo o agregar un repositorio existente.

1. Haz clic en **Crear un nuevo repositorio en tu disco duro...**



2. Para crear un repositorio nuevo, completa los campos:

**Create a New Repository** [X]

Name  
playground

Description  
For testing out GitHub Desktop.

Local Path  
/Users/steve/code/GitHub [Choose...]

☒ Initialize this repository with a README

Git Ignore  
None

License  
None

Cancel Create Repository

- "Name" (Nombre) define el nombre de tu repositorio a nivel local y en GitHub.
- "Description" (Descripción) es un campo opcional que puedes usar para brindar más información sobre el objetivo de tu repositorio.
- "Local path" (Ruta local) establece la ubicación de tu repositorio en tu computadora. De manera predeterminada, GitHub Desktop crea una carpeta GitHub en tu carpeta Documents (Documentos) para almacenar tus repositorios, pero puedes elegir cualquier ubicación en tu computadora. Tu nuevo repositorio será una carpeta dentro de la ubicación elegida. Por ejemplo, si colocas el nombre Tutorial a tu repositorio, se creará la carpeta Tutorial dentro de la carpeta que seleccionaste en tu ruta local. GitHub Desktop recuerda tu ubicación elegida la próxima vez que crees o clones un repositorio nuevo.
- Si inicializas este repositorio con un README (Léeme), se crea una confirmación inicial con un archivo README.md. README ayuda a las personas a comprender el objetivo de tu proyecto, por lo que recomendamos seleccionarlo y completarlo con información útil. Cuando alguien visita tu repositorio en GitHub, el archivo README es lo primero que verán a medida que aprenden sobre tu proyecto. Para obtener más información, consulta "Acerca de los archivos README".

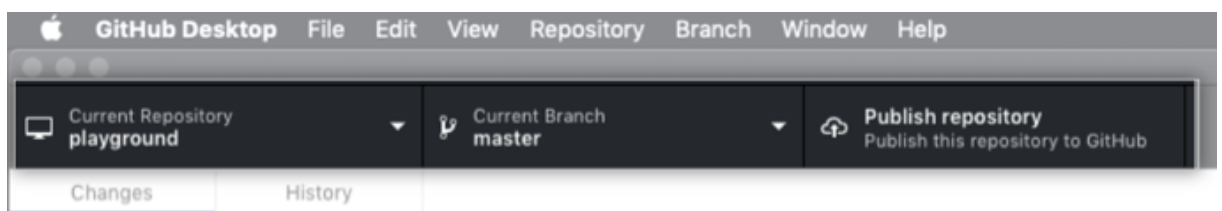
- e. El menú desplegable Git ignore (Ignorar Git) te permite agregar un archivo personalizado para ignorar los archivos específicos en tu repositorio local que no deseas almacenar en el control de la versión. Si existe un idioma o encuadre específico que estarás usando, puedes seleccionar una opción desde la lista disponible. Si recién estás comenzando, puedes omitir esta selección. Para obtener más información, consulta "Ignorar archivos".
- f. El menú desplegable License (Licencia) te permite agregar una licencia de código abierto para un archivo LICENSE (Licencia) en tu repositorio. No tienes que preocuparte por aprender cómo agregar una licencia inmediatamente. Para obtener más información sobre las licencias de código abierto disponibles y cómo agregarlas a tu repositorio, consulta "Licenciar un repositorio".

3. Haz clic en **Crear repositorio**.

### PASO 3. EXPLORAR GITHUB DESKTOP

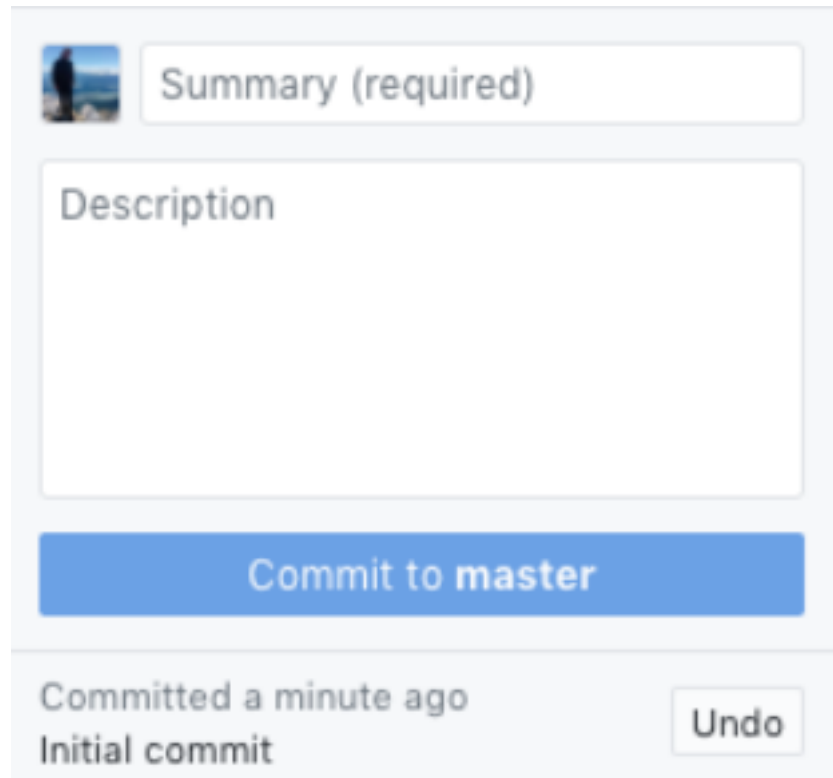
Ahora que has creado un repositorio, verás un menú de archivo en la parte superior de la pantalla. Allí es donde puedes acceder a la configuración y las acciones que puedes realizar en GitHub Desktop. La mayoría de las acciones tienen atajos del teclado para ayudarte a trabajar con más eficacia. Para conocer una lista completa de los atajos del teclado, consulta ["Atajos del teclado en GitHub Desktop"](#).

- 1. Debajo del menú se encuentra una barra que muestra el estado actual del repositorio en GitHub Desktop:
  - a. **Current repository** (Repositorio actual) muestra el nombre del repositorio en el que estás trabajando. Puedes hacer clic en **Current repository** (Repositorio actual) para alternar a un repositorio diferente en GitHub Desktop.
  - b. **Current branch** (Rama actual) muestra el nombre de la rama en la que estás trabajando. Puedes hacer clic en **Current branch** (Rama actual) para ver todas las ramas en tu repositorio, alternar a una rama diferente o crear una rama nueva. Una vez que creaste solicitud de extracción en tu repositorio, también puedes verlas haciendo clic en **Current branch** (Rama actual).
  - c. **Publish repository** (Publicar repositorio) aparece porque todavía no has publicado tu repositorio en GitHub. que harás a continuación en el próximo paso.

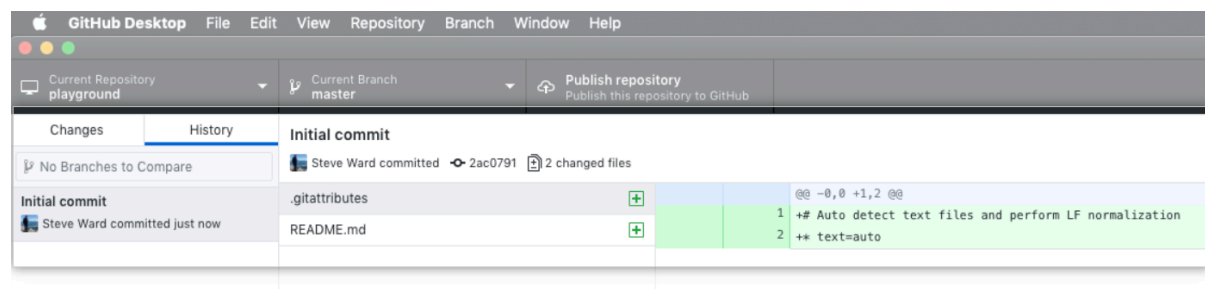


- 2. En la barra lateral a la izquierda, encontrarás la vista **Changes** (Cambios) y la vista **History** (Historial).

- a. La **vista Changes (Cambios)** muestra los cambios que realizaste a los archivos en tu rama actual pero aún no confirmaste en tu repositorio local. En la parte inferior, advertirás un cuadro con los cuadros de texto **"Summary" (Resumen)** y **"Description" (Descripción)** y un botón **Commit to master** (Confirmar en principal). Aquí es donde confirmarás los campos nuevos. El botón **Commit (Confirmar)** te permite saber en qué rama estás confirmando los cambios.



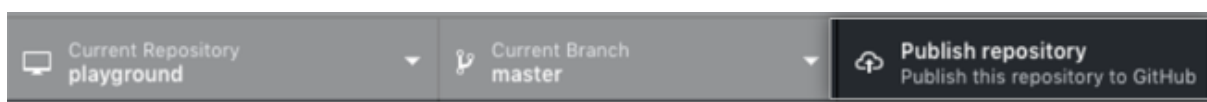
- b. La **vista History (Historial)** muestra las confirmaciones previas en la rama actual de tu repositorio. Deberías ver una "Initial commit" (Confirmación inicial) que fue creada por GitHub Desktop cuando creaste tu repositorio. A la derecha de la confirmación, según las opciones que seleccionaste al crear tu repositorio, es posible que veas los **archivos .gitattributes, .gitignore, LICENSE, o README**. Puedes hacer clic en cada archivo para ver una diferencia para ese archivo, que son los cambios realizados en el archivo en esa confirmación. La diferencia solo muestra las partes del archivo que han cambiado, no slo los contenidos completos del archivo.



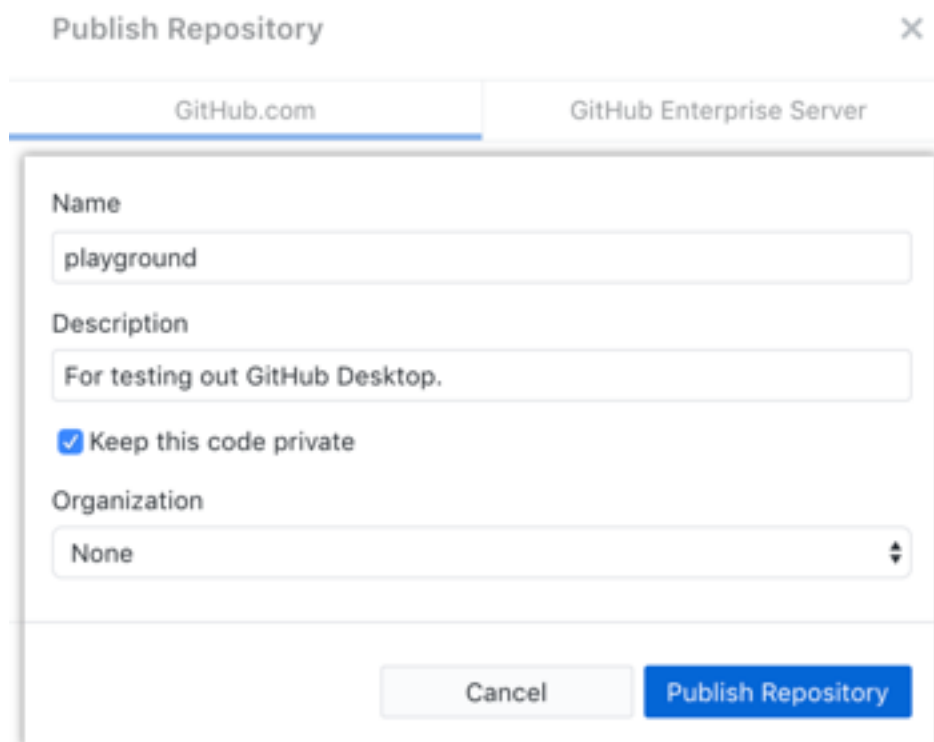
## PASO 4. SUBIR TU REPOSITORIO A GITHUB

Actualmente, tu repositorio solo existe en tu computadora, y eres el único que puede acceder al repositorio. Al publicar tu repositorio en GitHub se mantiene actualizado con múltiples computadoras y miembros del equipo en el mismo proyecto. Para publicar el repositorio, lo "subirás" a GitHub, lo que permite que también esté disponible en GitHub.com.

1. Haz clic en **Publish repository** (Publicar repositorio).



- o Verás algunos campos familiares. "Name" (Nombre) y "Description" (Descripción) coinciden con los campos completados cuando creaste el repositorio.
- o Verás la opción **Keep this code private** (Mantener la privacidad de este código). Selecciona esta opción si no deseas compartir tu código públicamente con otros usuarios en GitHub.
- o El desplegable **Organization** (Organización), si está presente, te permite publicar tu repositorio a una organización específica a la que perteneces en GitHub. No hay problemas si todavía no eres miembro de una organización.



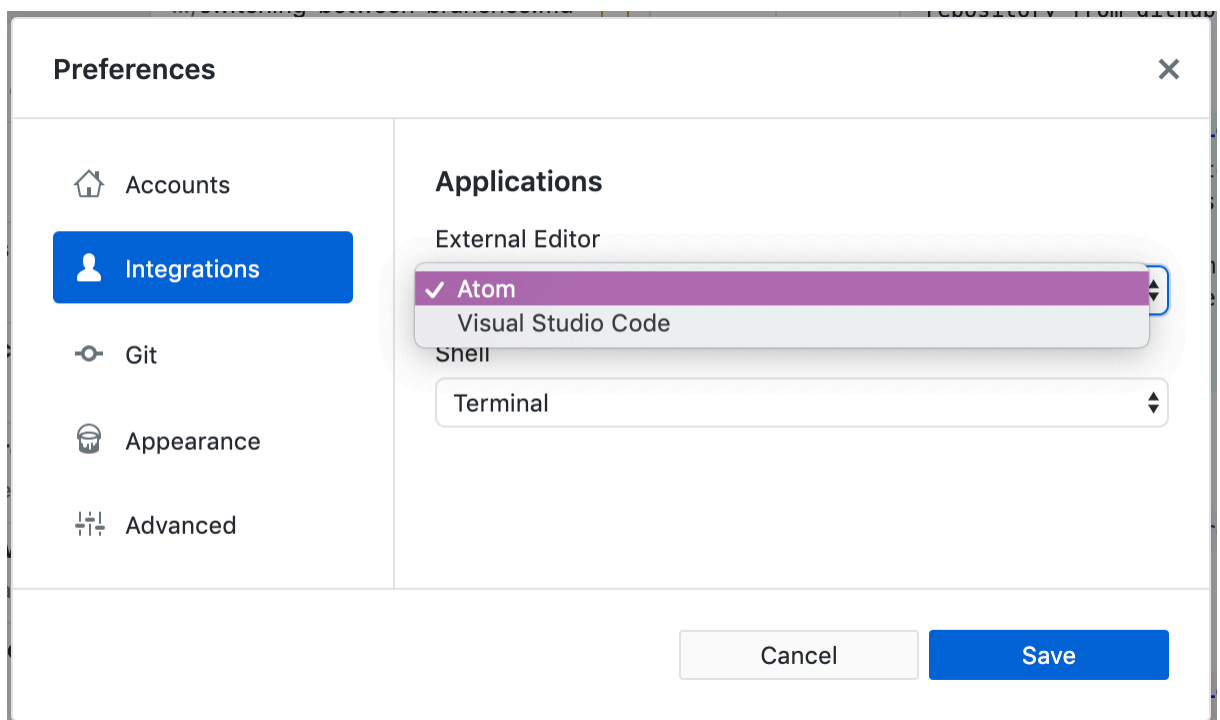
2. Haz clic en **Publish repository** (Publicar repositorio).
3. Puedes acceder al repositorio en GitHub.com desde el interior de GitHub Desktop. En el menú del archivo, haz clic en **Repository** (Repositorio), luego haz clic en **View on GitHub** (Ver en GitHub). Esto te llevará directamente hasta el repositorio en tu navegador predeterminado.

Ahora que el repositorio está publicado, volvamos a GitHub Desktop y realicemos más cambios a tu repositorio local. En primer lugar, repasaremos la configuración de un editor de texto predeterminado.

## PASO 5. CONFIGURAR UN EDITOR DE TEXTO

Para reducir la cantidad de tiempo dedicada a la configuración de tu entorno de desarrollo, lanzaremos un número de editores de texto y entornos de desarrollo integrados (IDE, por sus siglas en inglés) directamente desde GitHub Desktop. Desde un repositorio en GitHub Desktop, puedes abrir perfectamente la carpeta del proyecto en tu editor de texto favorito.

1. Haz clic en **File** (Archivo), luego haz clic en **Options** (Opciones) y luego haz clic en **Advanced** (Avanzado).
2. Usa el menú desplegable **External editor** (Editor externo) y selecciona un editor de la lista. Deberías ver los editores que has instalado en la lista. Si no ves ningún editor, instala un editor compatible como Atom. Para ver la lista de editores compatibles, consulta la integración "Open External Editor" (Abrir editor externo) en el repositorio GitHub Desktop.

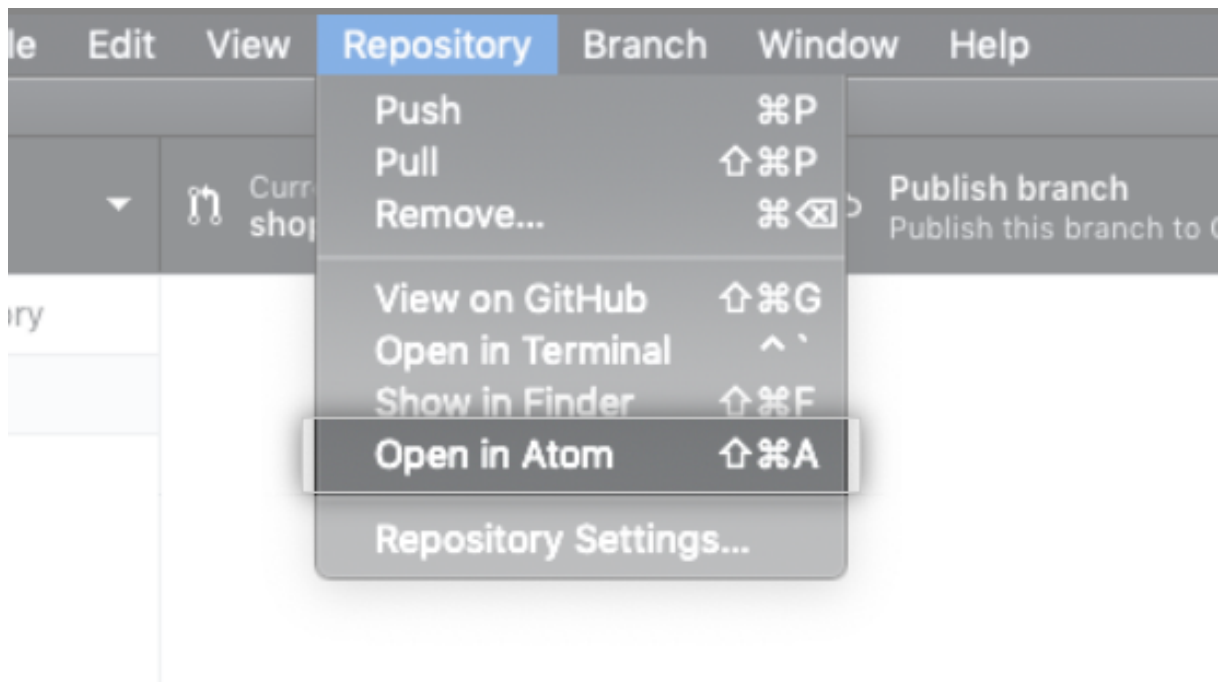


3. Si instalaste un editor nuevo, reinicia GitHub Desktop para que el editor esté disponible en el menú desplegable **External editor** (Editor externo).

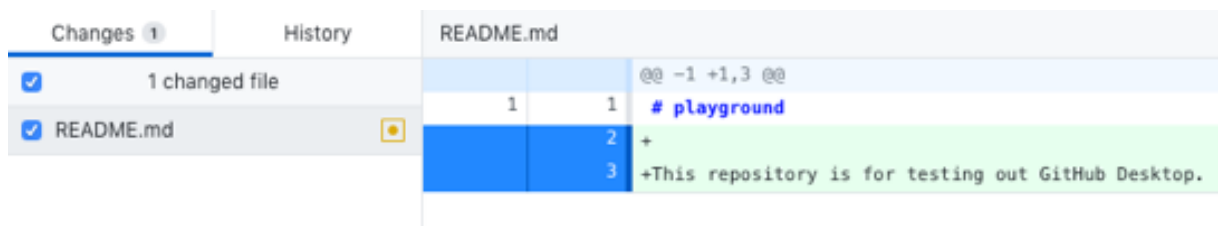
## PASO 6. REALIZAR, CONFIRMAR Y SUBIR CAMBIOS

Ahora que has configurado un editor predeterminado, estás listo para hacer cambios en tu proyecto y para comenzar a crear tu primera confirmación en tu repositorio.

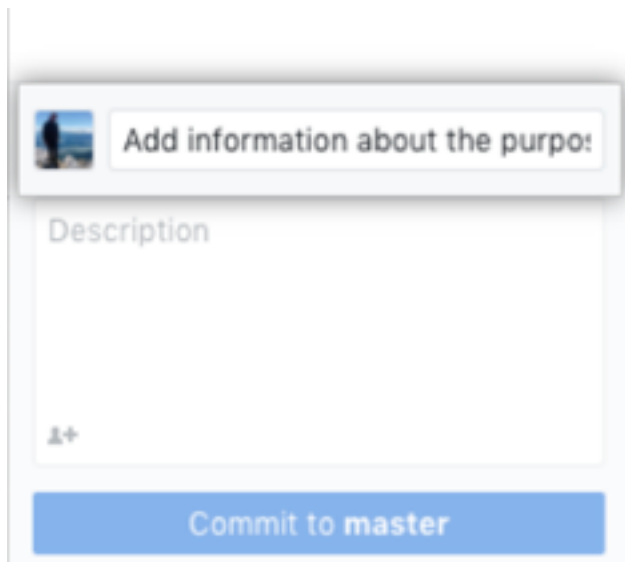
1. Para iniciar tu editor externo desde el interior GitHub Desktop, haz clic en **Repository** (Repositorio) y luego haz clic en **Open in EDITOR** (Abrir en EDITOR).



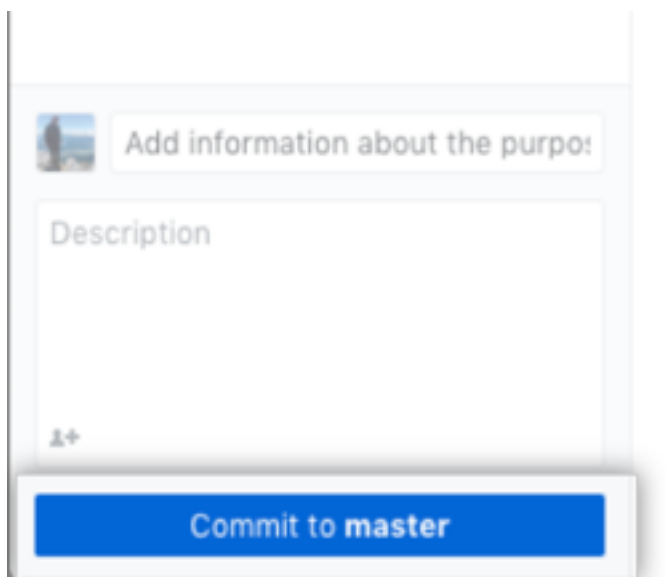
2. Comienza por realizar algunos cambios en el archivo README.md que creaste previamente. Agrega la información que describe el proyecto, como qué hace y por qué resulta útil. Recuerda que esta es la primera interacción que tendrán las personas con tu proyecto. Ahora estás listo para hacer tu primera confirmación.
3. Cambia desde el editor de texto hasta GitHub Desktop y desplázate hasta la pestaña **Changes** (Cambios). En la lista de archivos, deberías ver tu README.md. La marca de verificación junto al archivo README.md indica que los cambios que realizaste en el archivo formarán parte de la confirmación que realizaste. En el futuro, es posible que realices cambios a múltiples archivos pero solo quieras confirmar los cambios que realizaste en alguno de los archivos. GitHub Desktop te permite seleccionar los cambios específicos que deseas confirmar.



4. En la parte inferior de la lista **Changes** (Cambios), escribe un mensaje de confirmación. A la derecha de tu imagen de perfil, escribe una descripción breve de la confirmación. Dado que estamos cambiando el archivo README.md, "Agregar información sobre el objetivo del proyecto" sería un buen resumen de la confirmación. Debajo del resumen, verás un campo de texto "Description" (Descripción), donde puedes escribir una descripción más extensa de los cambios en la confirmación, que resulta útil al volver a mirar el historial de un proyecto y comprender por qué se hicieron los cambios. Dado que estás realizando una actualización básica de un archivo README.md, puedes omitir la descripción.



5. Haz clic en **Commit to master** (Confirmar en principal). El botón de confirmación muestra tu rama actual, que en este caso es master (principal), por lo que debes asegurarte de confirmar la rama que desees.



6. Para subir los cambios al repositorio remoto en GitHub, haz clic en **Push origin** (Subir origen).

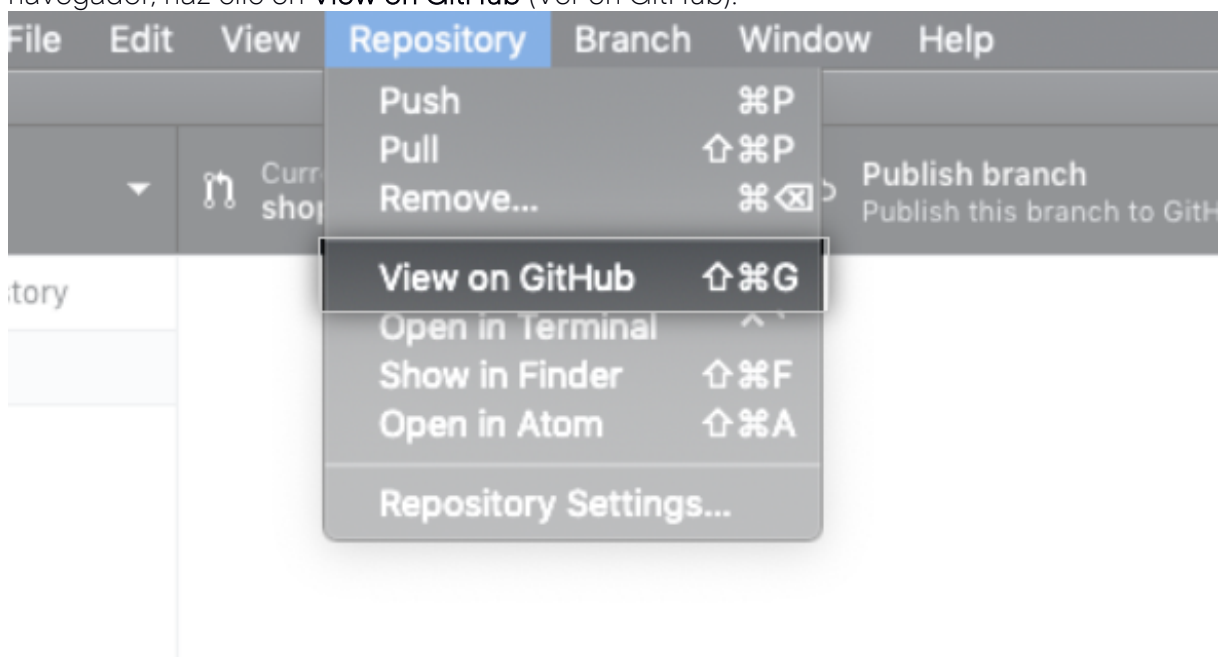


- ¿Recuerdas el botón **Publish** (Publicar) que usaste para publicar tu repositorio en GitHub? Ahora debería decir **Push origin** (Subir origen), con un código 1, junto al nombre, indicando que existe una confirmación que no ha sido subida a GitHub.
- El "origen" en **Push origin** (Subir origen) significa que estamos subiendo los cambios al remoto llamado origen (origen), que en este caso es tu repositorio del proyecto en GitHub.com. Hasta que hayas subido alguna de las nuevas confirmaciones GitHub, habrá diferencias entre el repositorio de tu proyecto en tu computadora y el repositorio del proyecto en GitHub.com.



Esto te permite trabajar localmente y solo subir tu trabajo a GitHub.com cuando estés listo.

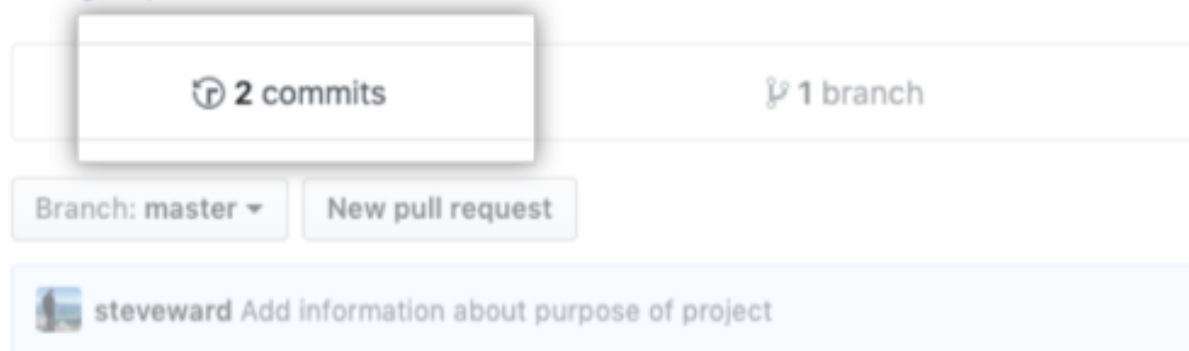
7. En el área abierta junto a la pestaña **Changes** (Cambios), verás sugerencias sobre lo que puedes hacer a continuación. Para abrir el repositorio en GitHub en tu navegador, haz clic en **View on GitHub** (Ver en GitHub).



8. En tu navegador, haz clic en **2 commits** (2 confirmaciones). Verás una lista de las confirmaciones en este repositorio en GitHub. La primera confirmación deberá ser la confirmación que acabas de realizar en GitHub Desktop.

For testing out GitHub Desktop.

Manage topics



## CONCLUSIÓN

¡Felicitaciones! Has creado un repositorio, publicado el repositorio en GitHub, realizado una confirmación y subido tus cambios. Esta es solo una muestra de todas las cosas que puedes hacer con GitHub y GitHub Desktop. Esperamos que este ejercicio te estimule para explorar un poco más.

