

# Guía instalación de Git y Github



## Qué es Git?

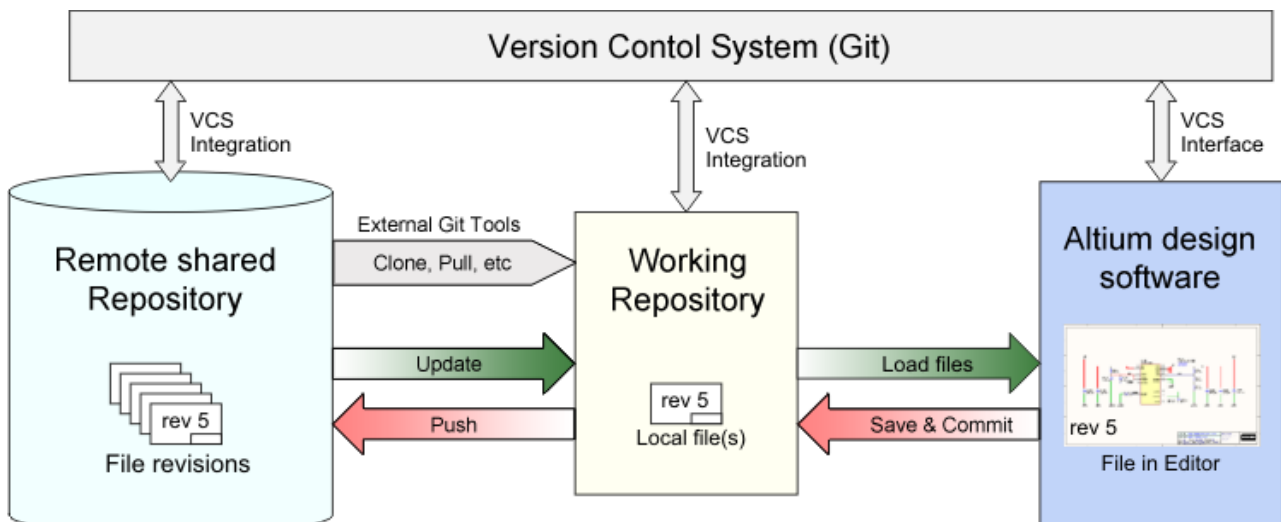
Git es un software de control de versiones gratis y de código abierto. Fue creado por Linus Torvalds en 2005. Esta herramienta es un sistema de control de versiones que fue inicialmente desarrollado para trabajar con varios desarrolladores en el núcleo de Linux.

Esto significa básicamente que Git es un rastreador de contenido. Así que Git puede ser utilizado para almacenar contenido — y se usa principalmente para almacenar código debido a otras características que proporciona.

Los proyectos de la vida real generalmente tienen múltiples desarrolladores trabajando en paralelo. Así que necesitan un sistema de control de versiones como Git para asegurarse de que no hay conflictos de código entre ellos.

Además, los requerimientos en este tipo de proyectos cambian constantemente. Así que un sistema de control de versiones permite a los desarrolladores revertir y regresar a una versión anterior de su código.

El sistema de ramas en Git permite a los desarrolladores trabajar individualmente en una tarea (Por ejemplo: una rama -> una tarea O una Rama -> un desarrollador). Básicamente, se puede pensar en Git como una aplicación de software pequeña que controla tu código base, si eres un desarrollador.



Muestra cómo funciona GitHub

## Repositorios Git

Si queremos empezar a usar Git, necesitamos saber dónde alojar nuestros repositorios.

Un repositorio (o "Repo" para abreviar) es un proyecto que contiene múltiples archivos. En nuestro caso un repositorio contendrá archivos basados en código.

Hay dos maneras en que puedes alojar tus repositorios. Uno es en línea (en la nube) y la segunda es fuera de línea (auto-instalado en tu servidor).

Hay tres servicios de alojamiento popular de Git: GitHub (propiedad de Microsoft), GitLab (propiedad de GitLab) y BitBucket. Usaremos GitHub como nuestro servicio de alojamiento.

## **Antes de usar Git debemos saber por qué lo necesitamos.**

### **Git facilita la contribución a proyectos de código abierto**

Casi todos los proyectos de código abierto utilizan GitHub para gestionar sus proyectos. Usar GitHub es gratis si tu proyecto es de código abierto, e incluye un wiki y un rastreador de problemas que facilita la inclusión de documentación más detallada y recibir retroalimentación sobre tu proyecto.

Si quieres contribuir, simplemente bifurcas (obtienes una copia de) un proyecto, realizas tus cambios, y luego envías un Pull Request al proyecto utilizando la interface web de GitHub. Este Pull Request es tu manera de decirle al proyecto que estás listo para que revisen tus cambios.

### **Documentación**

Utilizando GitHub, facilitas la obtención de excelente documentación. Su sección de ayuda y las guías tienen artículos para casi cualquier tema relacionado a Git en el que puedas pensar.

### **Opciones de Integración**

GitHub puede integrarse con plataformas comunes como Amazon y Google Cloud, con servicios como Code Climate para rastrear tus comentarios y puede resaltar la sintaxis en más de 200 lenguajes de programación diferentes.

### **Rastrea cambios en tu código a través de versiones**

Cuando varias personas colaboran en un proyecto, es difícil mantener el seguimiento de las revisiones — quién cambió qué, cuándo, y dónde están almacenados esos archivos.

GitHub se ocupa de este problema manteniendo un seguimiento de todos los cambios que se han enviado al repositorio.

Al igual que cuando se usa Microsoft Word o Google Drive, puedes tener un historial de las versiones de tu código, de manera que las versiones previas no se pierden con cada iteración. Es fácil regresar a la versión previa y contribuir a tu trabajo.

### **Muestra tu trabajo**

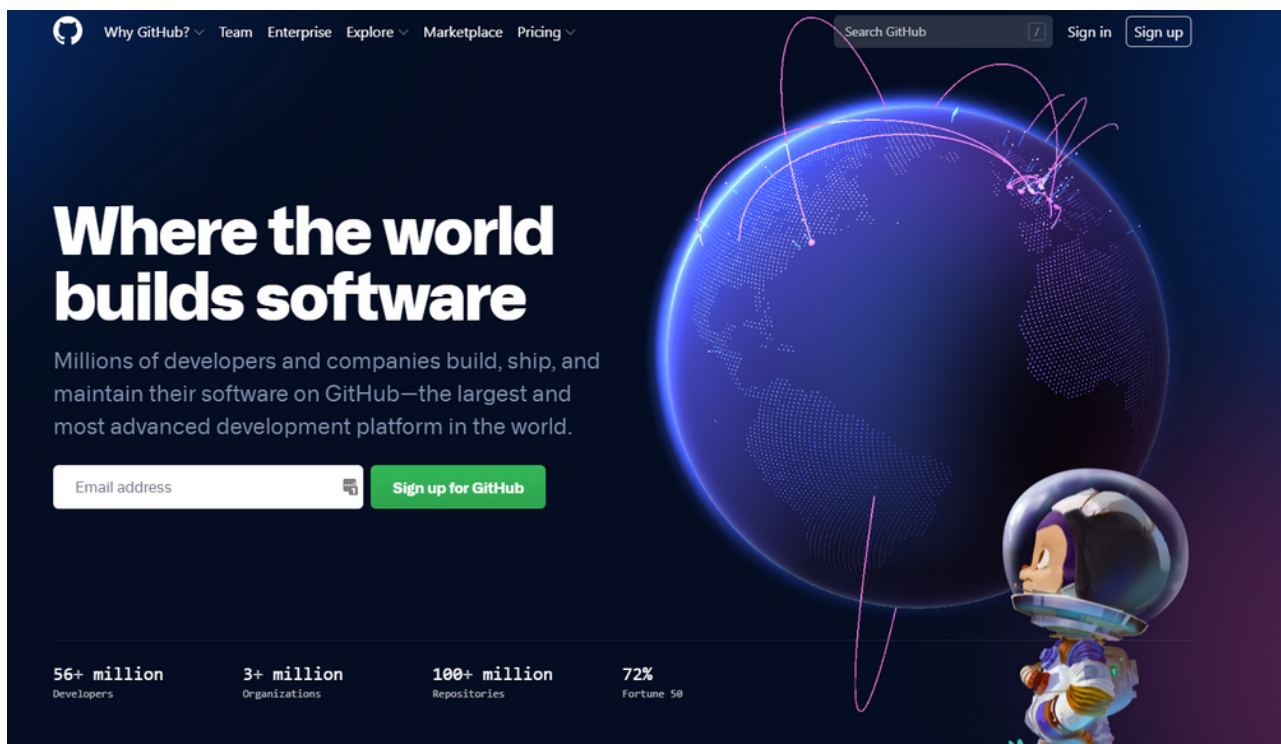
¿Eres un desarrollador que desea atraer a reclutadores? GitHub es la mejor herramienta en la que puedes confiar para esto.

Hoy, al buscar nuevos reclutas para sus proyectos, la mayoría de las compañías miran los perfiles de GitHub. Si tu perfil está disponible, tendrás mayores posibilidades de ser reclutado incluso si no eres de una gran universidad o colegio.

## **Ahora aprenderemos cómo usar Git y GitHub**

### **Creación de cuenta de GitHub**

Para crear tu cuenta, necesitas ir al sitio web de GitHub y llenar la forma de registro.



La página oficial de Github

## Instalación de Git

Ahora necesitamos instalar las herramientas de Git en nuestra computadora. Utilizaremos CLI (interfaz de línea de comandos) para comunicarnos con GitHub.

Para Ubuntu:

1. Primero, actualiza tus paquetes

```
sudo apt update
```

2. A continuación, instala Git y GitHub con apt-get

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

3. Finalmente, verifica que Git se instaló correctamente

```
git --version
```

4. Ejecuta los siguientes comandos con tu información para establecer un nombre de usuario y un correo electrónico predeterminados para cuando vayas a salvar tu trabajo.

```
git config --global user.name "MV Thanoshan"
git config --global user.email "example@mail.com"
```

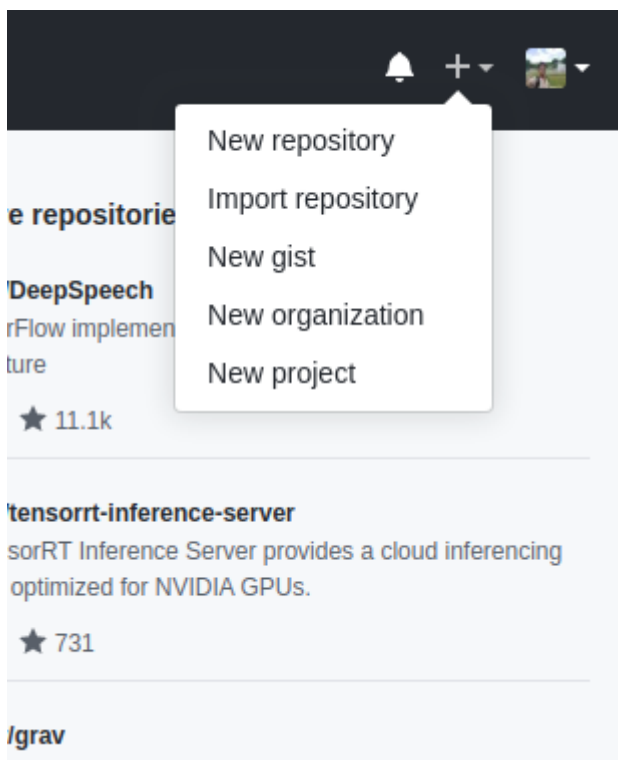
## Trabajando con proyectos GitHub

Trabajaremos con proyectos GitHub en dos maneras.

### Tipo 1: Crear el repositorio, clonarlo en tu PC y trabajar en él. (Recomendado)

El tipo 1 involucra la creación de un repositorio totalmente nuevo en GitHub, clonarlo en nuestra computadora, trabajar en nuestro proyecto y enviarlo de regreso.

Crea un nuevo repositorio haciendo clic en el botón de "Nuevo repositorio" en la página web de GitHub.





Elije un nombre para tu primer repositorio, agrega una pequeña descripción, marca la opción "Inicializar este repositorio con un README", y haz clic en el botón "Crear Repositorio".

## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner

Repository name \*

 ThanoshanMV / My-GitHub-Project 

Great repository names are short and memorable. Need inspiration? How about [stunning-octo-funicular?](#)

Description (optional)


This is my first project

- ☒ **Public**  
Anyone can see this repository. You choose who can commit.
- ☐ **Private**  
You choose who can see and commit to this repository.

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

☒ **Initialize this repository with a README**  
This will let you immediately clone the repository to your computer.

Add .gitignore: **None**

Add a license: **None** 

Create repository



Bien hecho! Tu primer repositorio de GitHub fue creado.

Tu primera misión es obtener una copia del repositorio en tu computadora. Para hacer eso, necesitas "clonar" el repositorio en tu computadora.

Clonar un repositorio significa que estás tomando un repositorio que está en el servidor y lo estás clonando a tu computadora – es lo mismo que descargarlo. En la página del repositorio, necesitas obtener la dirección "HTTPS".

Una vez que tienes la dirección del repositorio, necesitas utilizar tu terminal. Usa el siguiente comando en tu terminal. Cuando estés listo puedes ingresar esto:

```
git clone [DIRECCION HTTPS]
```

Este comando realizará una copia local del repositorio alojado en la dirección dada.

Mensaje de salida del comando "git clone"

Ahora, tu repositorio está en tu computadora. Necesitas moverte en él con el siguiente comando.

```
cd [NAME OF REPOSITORY]
```

Como puedes ver en la imagen de arriba, el nombre de mi repositorio es "My-GitHub-Project" y este comando me hizo ir al directorio específico.

Ahora, en ese folder podemos crear archivos, trabajar en ellos y guardarlos localmente. Para guardarlos en un lugar remoto — como GitHub — tenemos que hacer un proceso llamado "commit". Para hacer esto, regresa a tu terminal. Si la cerraste, como dije anteriormente, usa el comando 'cd'.

```
cd [NAME OF REPOSITORY]
```

Ahora, en la terminal, estás en el directorio de tu repositorio. Hay 4 pasos en un commit: 'status', 'add', 'commit' y 'push'. Todos los siguientes pasos deben ejecutarse dentro de tu proyecto. Repasemos uno por uno.

1. "status": La primer cosa que necesitas hacer es revisar los archivos que has modificado. Para hacer esto, puedes escribir el siguiente comando para hacer aparecer una lista de cambios.

```
git status
```

2. “add”: Con la ayuda de la lista de cambios, puedes agregar todos los archivos que quieras cargar con el siguiente comando,

```
git add [NOMBRE DE ARCHIVO] [NOMBRE DE ARCHIVO] [...]
```

En nuestro caso, agregaremos un archivo HTML simple.

```
git add sample.html
```

3. “commit”: Ahora que hemos agregado los archivos de nuestra elección, necesitamos escribir un mensaje para explicar lo que hemos hecho. Este mensaje puede ser útil después si queremos revisar el historial de cambios. Aquí hay un ejemplo de lo que podemos poner en nuestro caso.

```
git commit -m "Se agregó archivo HTML de muestra que contiene sintaxis basica"
```

4. “push”: Ahora podemos poner nuestro trabajo en GitHub. Para hacer eso, necesitamos 'enviar' nuestros archivos a Remote. Remote es una instancia duplicada de nuestro repositorio que vive en algún otro lugar en un servidor remoto. Para hacer esto, debemos saber el nombre del Remote (En general, Remote es nombrado origen). Para encontrar ese nombre, escribe el siguiente comando.

```
git remote
```

Como puedes ver en la imagen, dice que el nombre de nuestro remote es origen. Ahora podemos 'enviar' de manera segura nuestro trabajo con el siguiente comando.

```
git push origin master
```

Ahora, si vamos a nuestro repositorio en la página web de GitHub, podemos ver el archivo sample.html que hemos enviado a Remote — GitHub!

**NOTA:** A veces cuando estás usando comandos Git en la terminal, te puede llevar al editor de texto VIM (un editor de texto basado en CLI). Para deshacerte de esto, tienes que escribir

```
:q
```

y presionar ENTER.

Describes how pull & push work

Pulling es el acto de recibir de GitHub.

Pushing es el acto de enviar a GitHub.

## Tipo 2: Trabajar en tu proyecto localmente y después crear el repositorio en GitHub y enviarlo a remote.

El tipo 2 permite hacer un nuevo repositorio de un folder existente en nuestra computadora y enviarlo a GitHub. En muchos casos es posible que ya hayas creado algo en tu computadora y repentinamente quieras convertirlo en un repositorio en GitHub.

Voy a explicarte esto con un proyecto web de formulario de encuesta que hice antes y que no fue agregado a GitHub.

Como ya mencioné, cuando se ejecuta cualquier comando Git, tenemos que asegurarnos que estamos en el directorio correcto en la terminal.

De manera predeterminada, cualquier directorio en nuestra computadora no es un repositorio Git – pero lo podemos convertir en un repositorio Git ejecutando el siguiente comando en la terminal.

```
git init
```

Después de convertir nuestro directorio en un repositorio Git, la primera cosa que necesitamos hacer es revisar los archivos que tenemos utilizando el siguiente comando.

```
git status
```

Así que hay dos archivos en ese directorio que necesitamos "agregar" a nuestro Repo.

```
git add [FILENAME] [FILENAME] [...]
```

**NOTA:** Para "agregar" todos los archivos en nuestro repositorio podemos usar el siguiente comando.

```
git add .
```

Después de que el área de preparación (el proceso de Add) está completo, podemos revisar si nuestros archivos fueron agregados satisfactoriamente o no ejecutando `git status`

Si esos archivos en particular están en verde como la imagen de abajo, haz hecho tu trabajo!

Después tenemos que hacer "commit" con una descripción.



```
git commit -m "Agregando un formulario web de encuesta"
```

Si mi repositorio inició en GitHub y lo descargo a mi computadora, un remote ya estará unido a él (Tipo 1). Pero si estoy iniciando mi repositorio en mi computadora, no tiene un remote asociado con él, así que necesito agregarlo (Tipo 2).

Así que para agregar ese remote, primero tenemos que ir a GitHub. Crear un nuevo repositorio y nombrarlo como queramos almacenarlo en GitHub. Después hacer clic en el botón "Crear repositorio".

**NOTA:** En el tipo 2, por favor no inicialices el repositorio con un archivo README cuando crees un nuevo repositorio en la página web de GitHub.

Después de hacer clic en el botón "Crear repositorio" encontrarás la siguiente imagen como una página web.

Copia la dirección HTTPS. Ahora crearemos el remote para nuestro repositorio.

```
git remote add origin [DIRECCION HTTPS]
```

Después de ejecutar este comando, podemos revisar si hemos agregado el remote satisfactoriamente o no con el siguiente comando

```
git remote
```

Y si genera "origin", haz agregado el remote a tu proyecto.

**NOTA:** Solo recuerda que podemos indicar cualquier nombre para el remote cambiando el nombre "origin". Por ejemplo:

```
git remote add [NOMBRE REMOTE] [DIRECCION HTTPS]
```

Ahora, podemos enviar nuestro proyecto a GitHub sin ningún problema!

```
git push origin master
```

Después de completar estos pasos uno por uno, si vas GitHub puedes encontrar tu repositorio con los archivos!

## Conclusión

Muchas gracias a todos por leer. Expliqué lo básico de Git y GitHub. Te animo fuertemente a leer más artículos relacionados a Git y GitHub. Espero que este artículo te haya ayudado.

Gracias.

**¡Feliz Programación!**

Traducido del artículo de [Thanoshan MV](#) - [The beginner's guide to Git & GitHub](#)

---

Aprende a codificar de forma gratuita. El plan de estudios de código abierto de freeCodeCamp ha ayudado a más de 40,000 personas a obtener trabajos como desarrolladores. [Empezar](#)