



# INTRODUCCIÓN A GIT Y GITHUB: GIT COMO HERRAMIENTA DE INVESTIGACIÓN COLABORATIVA

**Miguel Jafert Serrano Mantilla**

*Universidad Industrial De Santander*

*Grupo De Óptica Y Tratamiento De Señales*

# Índice



**Resumen**

En el ámbito científico es bastante común trabajar en colaboración con una o más personas en un mismo proyecto, de esta forma se hace necesario un manejo óptimo de las herramientas compartidas para la investigación.

A la hora de trabajar en colaboraciones se deberán añadir, remover , modificar múltiples archivos entre todos los miembros sin que ninguna de estas acciones afecte de manera negativa a algún compañero. Git surge como ayuda para esto, es un software que nos permite tener un repositorio global en el cual cada usuario puede tener una ramificación de este, modificarlo a su gusto sin afectar las versiones de otros. Así, una vez sea necesario unir todos los avances y modificaciones a la rama principal, avanzando en los distintos proyectos sin modificar el trabajo de otros o eliminándolo. En caso de afectar al trabajo de alguien más, Git permite volver atrás en el tiempo, a como era el repositorio antes de las modificaciones y recuperar lo perdido.

# 1. Introducción

## 1.1. Git

En 1991 el kernel de Linux realizaba sus mantenimiento utilizando archivos y parches, algo similar a lo que hacen actualmente muchos vídeojuegos, hasta que en 2002 se empezó a utilizar un control de versiones llamado BitKeeper lo cual permitía actualizar el kernel de Linux de forma más organizada y controlada pero en 2005 este software dejó de ser gratuito, debido a esto Linus Torvals, el mismo creador de Linux diseñó Git. Fue diseñado como una alternativa rápida y eficiente a los sistemas de control de versiones existentes, y desde entonces se ha convertido en uno de los sistemas de control de versiones más utilizados en el mundo.

Git permite a los desarrolladores llevar un registro de los cambios en su código con el tiempo, así como colaborar con otros en el mismo código base. Funciona almacenando instantáneas del código en diferentes puntos en el tiempo, que se pueden comparar, fusionar y revisar según sea necesario. Git también admite ramificaciones, lo que permite a los desarrolladores trabajar en diferentes partes del código simultáneamente sin afectar el código base principal.

Una de las características clave de Git es su arquitectura distribuida. Esto significa que cada usuario tiene una copia completa del código y su historial, y puede trabajar de forma independiente sin depender de un servidor central. Esto facilita el trabajo sin conexión, así como la copia de seguridad y el intercambio de código.

Git también es software de código abierto, lo que significa que es gratuito para usar y el código fuente está disponible para que cualquiera lo inspeccione y modifique. Esto ha llevado a una gran y activa comunidad de usuarios y contribuyentes, quienes han contribuido con numerosas funciones e mejoras al software a lo largo de los años.

### 1.2. Github

GitHub es una plataforma web que se basa en Git y brinda herramientas para colaboración y gestión de proyectos de software. Además de alojar repositorios de Git, GitHub proporciona una amplia gama de funciones, incluidas herramientas para el seguimiento de problemas, discusión en equipo, revisión de código, integración continua y más. GitHub también ofrece planes de pago que incluyen características adicionales, como privacidad y control de acceso, para equipos y organizaciones.

El plan gratuito de GitHub ofrece 1 GB de almacenamiento para repositorios públicos y un ancho de banda ilimitado. Para repositorios privados, la cuenta gratuita ofrece un límite de hasta 3 GB de almacenamiento adicional y un ancho de banda limitado. El almacenamiento gratuito de GitHub se renueva automáticamente cada mes. Esto significa que cada mes, la cantidad de almacenamiento disponible en la cuenta gratuita se reinicia a 1 GB para repositorios públicos y 3 GB para repositorios privados.

## 2. Instalación

### 2.1. Para usuarios de Windows

Para instalar Git en Windows solo se debe descargar e instalar el paquete de instalación desde la página oficial de Git: `https://git-scm.com/downloads`. El proceso de instalación es muy similar al de cualquier otro software en Windows:

1. Descargue el archivo de instalación de Git desde la página web de Git.
2. Ejecute el archivo de instalación y sigue las instrucciones en la pantalla para completar la instalación.

3. Verifica que Git está correctamente instalado abriendo la línea de comandos , cmd, y escribiendo:

```
git --version.
```

Si se desea, desde la misma página de git se puede descargar github desktop, el cual es un software que permite manipular git sin el uso de una ventana de comandas, sino con el uso de un software a base de ventanas.

## 2.2. Para usuarios de linux

Puedes instalar Git en Linux mediante los siguientes comandos en una terminal:

- Para sistemas basados en Debian (Ubuntu, Debian, etc.):

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

- Para sistemas basados en Red Hat (Fedora, CentOS, etc.):

```
sudo yum update
sudo yum install git
```

- Para sistemas basados en Arch Linux:

```
sudo pacman -Syu
sudo pacman -S git
```

Una vez que Git está instalado, puedes verificar la versión instalada con el comando:

```
git --version
```

## 2.3. Para usuarios de MacOs

Para instalar Git en Mac OS , se puede hacer descargando e instalando el paquete de instalación desde la página oficial de Git: `https://git-scm.com/downloads`. También se puede usar el gestor de paquetes Homebrew para instalar Git en Mac OS. Para instalarlo con Homebrew, debe seguir los siguientes pasos:

1. Abra la terminal de Mac OS.
2. Ejecute el siguiente comando para instalar Homebrew:

```
/bin/bash -c "$(curl -fsSL https:
//raw.githubusercontent.com/
Homebrew/install/master/install.sh)"
```

3. Una vez que se haya instalado Homebrew, ejecute el siguiente comando para instalar Git:

```
brew install git
```

4. Verifique que Git está correctamente instalado abriendo la Terminal y escribiendo:

```
git --version
```

# 3. Parte 1: Uso conjunto de Git y Github para uso personal

## 3.1. Uso como repositorio o lugar de almacenamiento

Github nos sirve como ese lugar de almacenamiento online de los repositorios hechos en git , si queremos hacer nuestro propio repositorio o trabajar o descargar alguno ya existente debemos clonarlo pero antes de hacer esto, hay que crearse una cuenta de Github. Para esto primero debe crearse una cuenta en github, para eso, puede hacer lo siguiente:

1. Vaya a el sitio web de GitHub: `https://github.com/`
2. Haga clic en el botón "Sign up/ Registrarse.[en] la esquina superior derecha de la página.
3. Rellene el formulario de registro con su nombre de usuario, dirección de correo electrónico y contraseña.
4. Verifique su dirección de correo electrónico haciendo clic en el enlace que envía GitHub.
5. Iniciar sesión

Parar configurar la cuenta de git con el computador es necesario aplicar los siguientes comandos:

```
git config --global user.name   Su   nombre
git config -- global user.email    su   correo de github
```

A veces cuando hagan comandos de Git les volverá a pedir user, en caso de que eso suceda, deben poner los comandos anteriormente mencionados y agregar el siguiente:

```
git config -- global credential.helper cache
```

Para iniciar un repositorio se pueden hacer dos cosas, o clonar uno ya existente o empezar uno. Para iniciar uno, directamente en la terminal de comandos, escribimos:

```
git init
```

Si se quiere clonar, es decir, descargar directamente al computador un repositorio ya existente para trabajar en él y editarlo desde el computador vamos a `github.com` buscamos el repositorio y donde dice <> *Code* copiamos el link que ahí aparece:

Para corroborar el estado en el que se encuentra el repositorio utilizamos :

```
git status
```

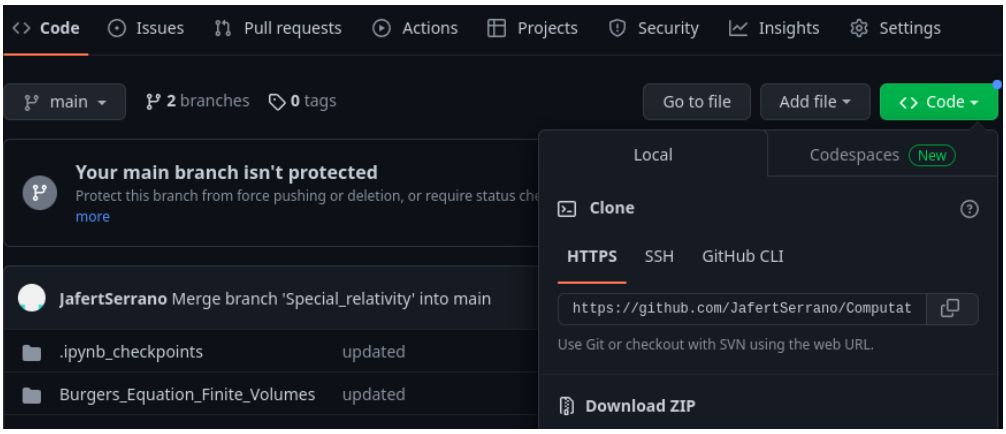


Figura 1: Ubicación del link del repositorio para clonar

Los comandos para clonar el repositorio son los siguientes:

```
git clone link Del Repo
git pull
```

Una vez realizado todo esto, ya podemos modificar cuantos archivos querramos pero no habremos modificado el repositorio de github este seguirá como si nada, entonces para ver qué está pasando utilizamos el comando status y podremos aplicar los siguientes comandos:

- Para agregar archivos.

  ```
  git add Archivo(s)
  ```

- Para remover archivos.

  ```
  git rm Archivo(s)
  ```

- Para asesinar archivos.

  ```
  git rm -f Archivo(s)
  ```

Una vez hecho cambios es necesario comentarlos, al hacerlo toma una captura de cómo se encuentra el repositorio en ese momento y lo guarda, para comentar se hace usando

```
git commit -m "Comentario"
```

## 3.2. Branches o ramificaciones

Aunque estemos trabajando solos surge la necesidad de modificar un archivo, bien podría ser un código en algún lenguaje o un archivo de texto pero sin alterar el original, esto ya sea para comparar

un método o modificaciones y tener así criterio alguno para escoger alguna de las versiones que se haga del archivo original, para esto, Git utiliza branches, es decir, ramifiaciones de un repositorio original, en los cuales podemos hacer tantas modificaciones como nosotros querramos sin alterar el contenido original.

- Para visualizar las branches o ramificaciones existentes se usa el comando.

```
git branch
```

- Para crear una nueva rama se puede hacer de dos formas:

```
git branch "Nombre de la rama"
git checkout -b "Nombre de la rama"
```

- Para borrar un rama se usa:

```
git branch -d "Nombre de la rama"
```

- Para cambiar de una rama a otra:

```
git checkout "Nombre de la rama"
```

Podemos, de cierta forma, mirar hacia atrás en el tiempo con Git a través del comando:

```
git log
```

El cual nos mostrará algo así:

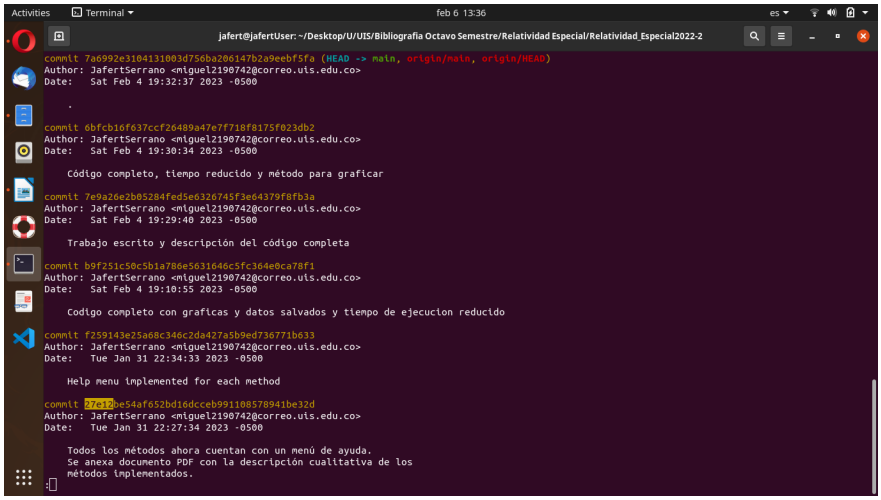


Figura 2: Historial de modificaciones a un repositorio.

Podemos volver a alguna de las versiones anteriores utilizando

```
git checkout "Primeros cuatro digitos del historial"
```

Esto nos cambia a una rama TEMPORAL, en la cual se encontraba el repositorio en ese instante, si cambiamos a la rama principal esta rama pasada quedará eliminada, si queremos hacerla permanente debemos utilizar el siguiente comando:

```
git switch "nombre de la rama"
```

Hay mejores formas de ilustrado el pasado de las modificaciones en las ramas de los repositorios, para esto se usa algo llamado grafos de Git o diagramas de árbol de git, vamos a ver dos ejemplos, para esto ejecutemos los dos siguientes comandos que me crean dos métodos nuevos de Git, *lg* y *lg*2:

```
git config --global alias.lg
"log --color --graph --
pretty=format:'%Cred%h%Creset
-%C(yellow)%d%Creset %s %Cgreen(%cr)
%C(bold blue)<%an>%Creset' --abbrev-commit"

git config --global alias.lg2
"log --graph --abbrev-commit --decorate --
format=format:'%C(bold blue)%h%C(reset)
- %C(bold cyan)%aD%C(reset)
%C(bold green)(%ar)%C(reset)%C(bold yellow)%d%C(reset)%n''
%s%C(reset) %C(dim white)- %an%C(reset)' --all"
```

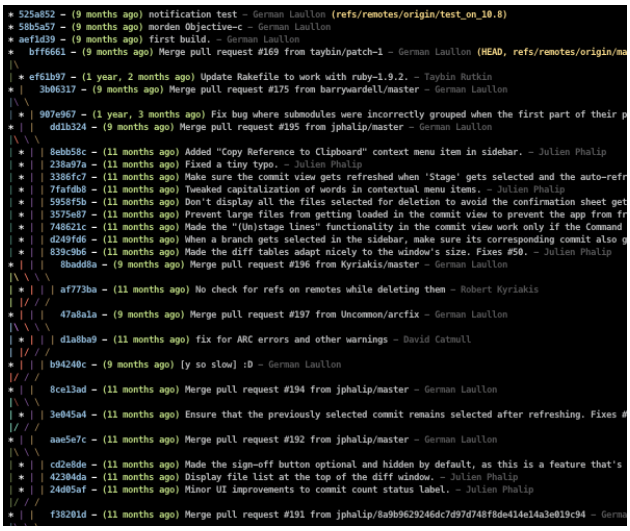

Figura 3: Visualización de los grafos de git con el método lg.

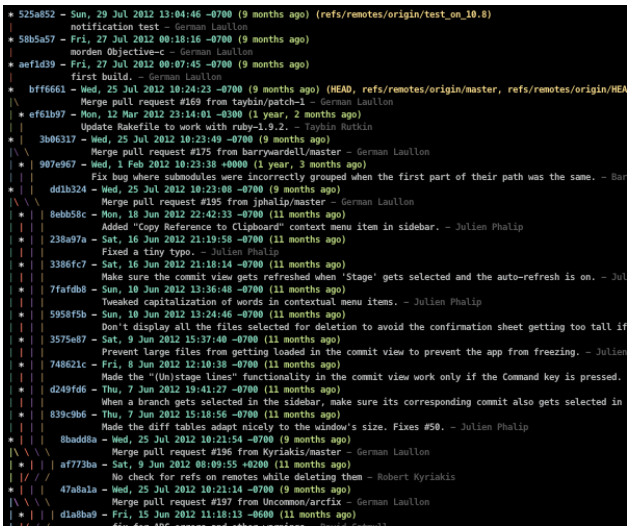


Figura 4: Visualización de los grafos de git con el método lg2.

Para más información sobre estos grafos se puede consultar `https://stackoverflow.com/questions/1057564/pretty-git-branch-graphs`.

Por último, si los cambios realizados a los archivos son satisfactorios y se desea alterar los originales, se deben unir los repositorios, para esto, ubicándonos en la rama que queremos modificar hacemos un merge, es decir:

```
git checkout "Rama a modificar"
git merge "Rama con la cual se quiere unir"
```

Esta operación solo me altera la rama en la que estamos dejando intactas las otras.

# 4. Parte 2:

## 4.1. Uso conjunto de git y github en colaboraciones

Esta parte consta de hacer lo mismo que lo anterior pero ver qué ocurre cuando una o más personas tratan de hacer merge desde dos ramas distintas a otra, bien puede ser la rama principal u otra.

Para ejemplificiar esto vamos a clonar dos veces en dos carpetas distintas un mismo repositorio para esto hacemos:

```
git clone link "nombre 1"
git clone link "nombre 2"
```

Y para dar la posibilidad de que una o más personas pueda hacer montajes al github ejecutamos lo siguiente:

```
git push   set  -upstream origin "nombre n"
```

Desde la terminal como desde github podemos controlar desde qué rama a qué rama queremos hacer modificaciones. Una vez querramos hacer una modificaciones vamos a pull requests y ahí creamos una petición, la cual se muestra en la figure 5:

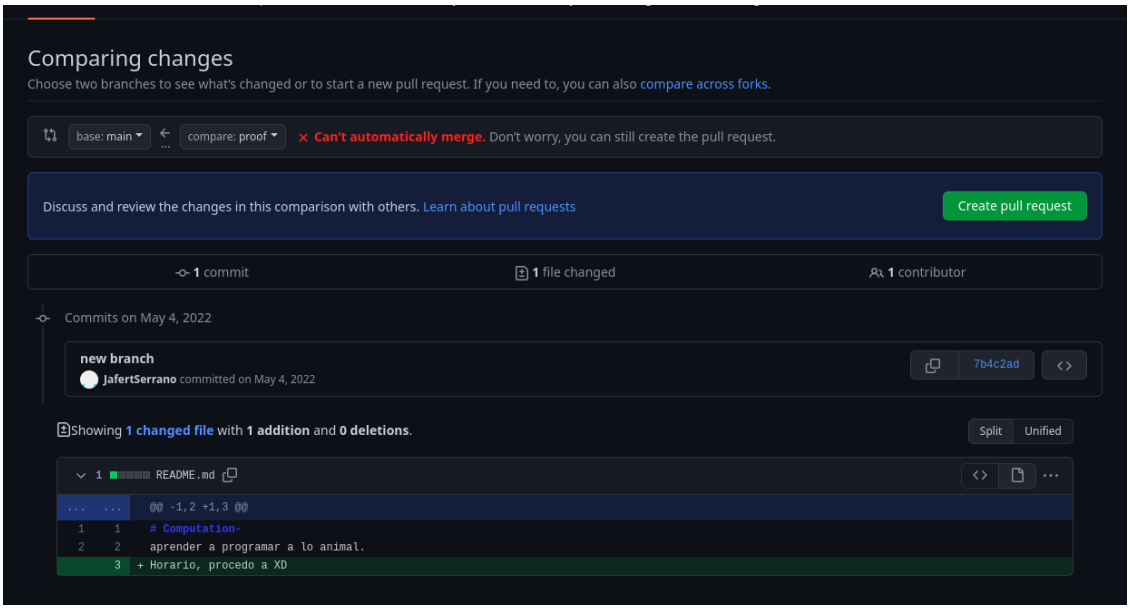


Figura 5: Pull request desde github.

Si no hay problemas con el archivo que se va a cambiar, github dará vía libre para hacer un commit y luego un merge pero si los archivos se modifican y entran en conflicto, saldrá la siguiente notificación mostrada en la figure 6:

Cuando le demos resolver conflictos nos mostrará los cmabios realizados en el archivo y nos pedirá escoger qué cambios mantener y cuaĺes eliminar, una vez se haga la modificación permitirá hacer commits y merge.

## 4.2. Github como free host para páginas web

GitHub Pages es un servicio de alojamiento de sitios web estáticos que permite alojar un sitio web directamente desde un repositorio de GitHub. Para utilizarlo hay que seguir los siguientes pasos

1. Crear un repositorio: Vaya a GitHub y crea un nuevo repositorio para tu sitio web. El nombre del repositorio debe tener el formato username.github.io, donde username es su nombre de usuario de GitHub.

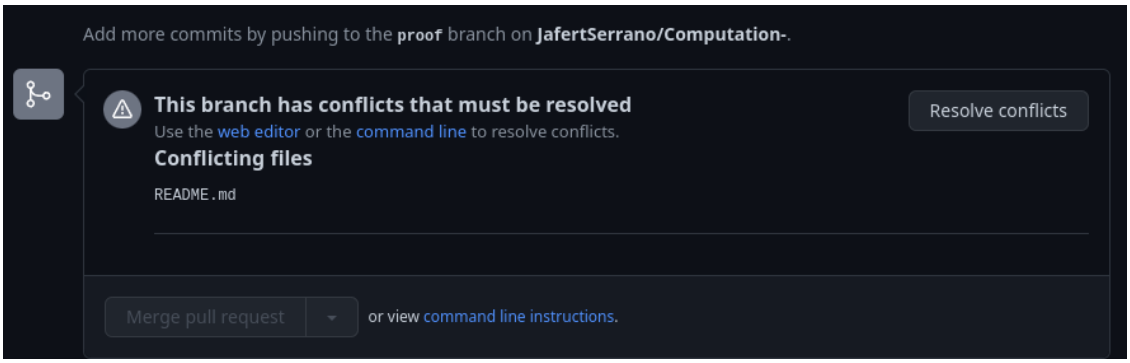


Figura 6: Notificación de conflicto a la hora de modificar un archivo.

2. Elegir un tema: Puede elegir un tema para tu sitio web yendo a la configuración del repositorio y seleccionando la sección "GitHub Pages". Puede elegir entre varios temas predefinidos o usar un tema personalizado seleccionando "Tema personalizado".

3. Agregar su contenido: Puedes agregar tu contenido al repositorio usando markdown, HTML o un generador de sitios web estáticos como Jekyll, al igual que en todo host, el nombre del archivo debe ser index, en este caso usaremos la extensión .html, es decir la página web deberá llamarse ïndex.html".

4. Publicar el sitio web: Una vez que hayas agregado todo el contenido, pueden publicar su sitio web empujando los cambios al repositorio de GitHub. El sitio web estará disponible en https://username.github.io, donde username es suu nombre de usuario de GitHub.

5. Hay que tener en cuenta que GitHub Pages está diseñado para alojar sitios web estáticos y no admite scripts de lado del servidor ni contenido dinámico. Si necesita construir un sitio web más complejo, es posible que desee considerar otras opciones de alojamiento.