Manual de Git

Introducción al control de versiones e historia de Git

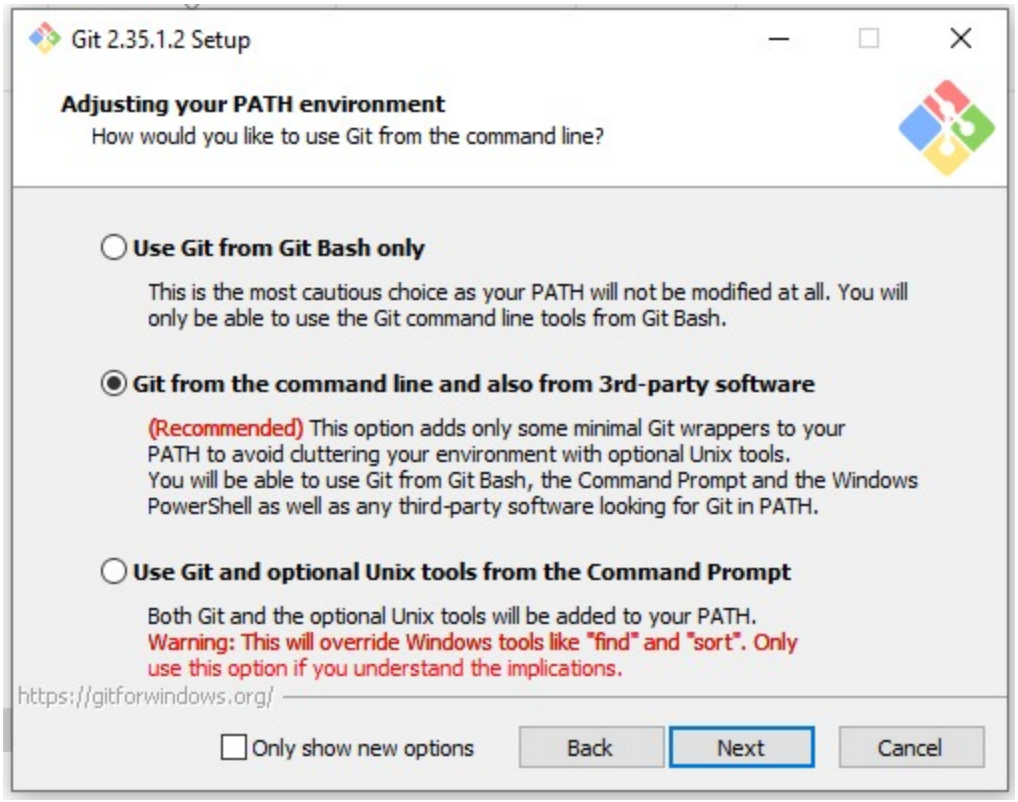
Git fue creado por Linus Torvalds en el año 2005, este es un sistema que permite controlar las versiones, esto quiere decir que nos ayuda a llevar un registro de los cambios realizados en archivos o documentos. Este fue creado para que los desarrolladores guarden el código de sus proyectos y herramientas pero actualmente es usado para que otros usuarios guarden cualquier tipo de información.

Instalación de Git para sistema Windows

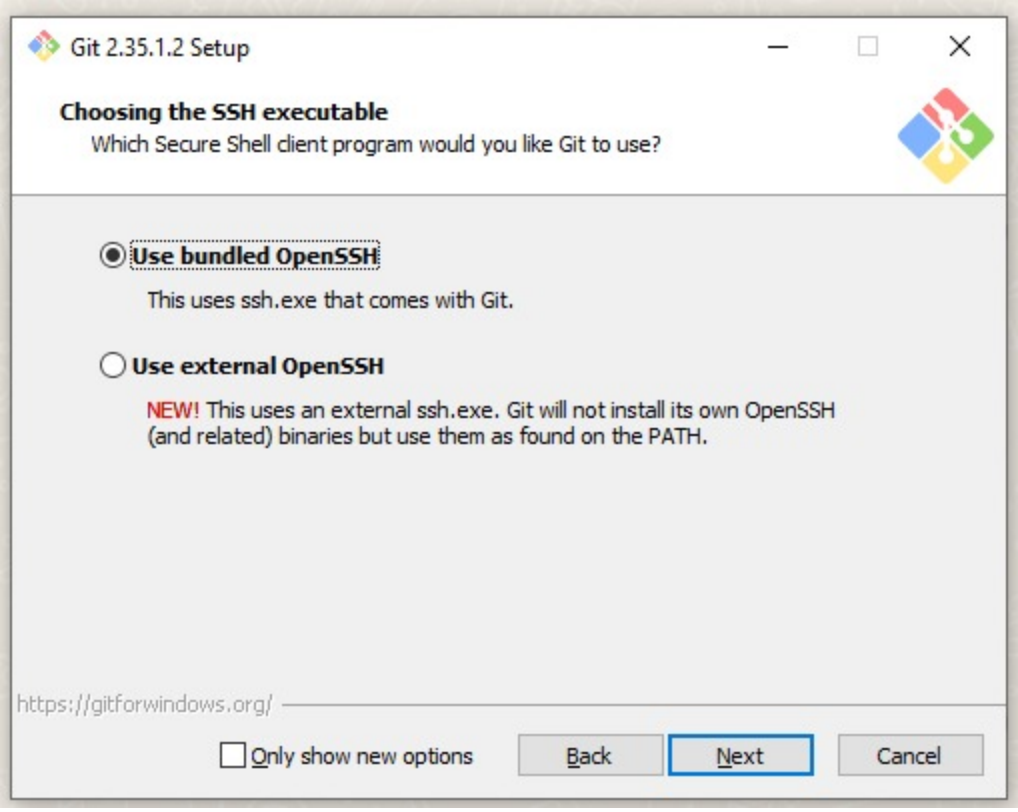
Para instalar Git en Windows, vaya al siguiente enlace: http://git-scm.com/download/win seguidamente dar clic en "**Click here download**" y la descarga empezará automáticamente.

Luego de que se haya descargado abrimos el archivo dando doble clic, aparecerá una ventana preguntando "*¿Quieres permitir que esta aplicación realice cambios en tu dispositivo?*" Elija la opción **sí,** en el proceso de instalación se deja por defecto las opciones dando clic en Next, pero tenga en cuenta que en los siguientes casos marcar las siguientes opciones para permitir usar Git desde la línea de comando y elegir la herramienta SSH que está incluida en este programa.

Para ejecutar Git desde la línea de comando del sistema operativo se selecciona esta opción:



Y selecciona esta opción para usar la herramienta clave SSH que le permite iniciar sesión en su servidor sin necesidad de una contraseña:



Creación de cuenta de Git y GitHub

Hasta ahora se ha visto la historia y el como instalarlo, en esta parte veremos como se podrá crear una cuenta o personalizar el funcionamiento de git.

Se presentara algunos de sus principales ajustes, con estas opciones es fácil trabajar como se desea

Configuración git

Una de las primeras acciones en git es poder configurar nuestro nombre de usuario y el email

→Para usar estos comando abrimos Git bash ó el símbolo del sistema (CMD)

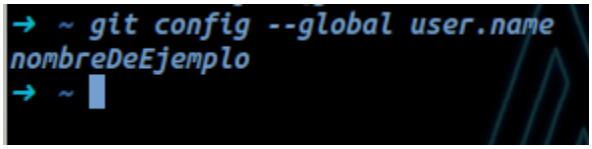
```
git config --global user.name '<nombre_deseado>'
```

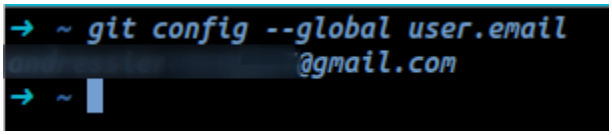
```
git config --global user.email ejemplo@mail.com
```

Para saber el usuario o email ejecutamos los siguientes comandos

```
git config --global user.name
```

```
git config --global user.email
```



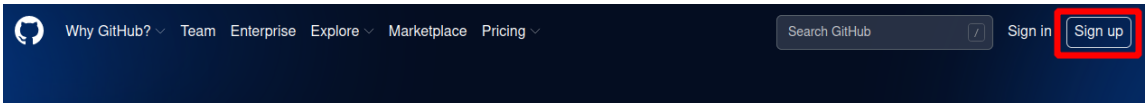


para ver toda la configuración de git ejecutamos `git config --list` , nos aparecerá los siguiente

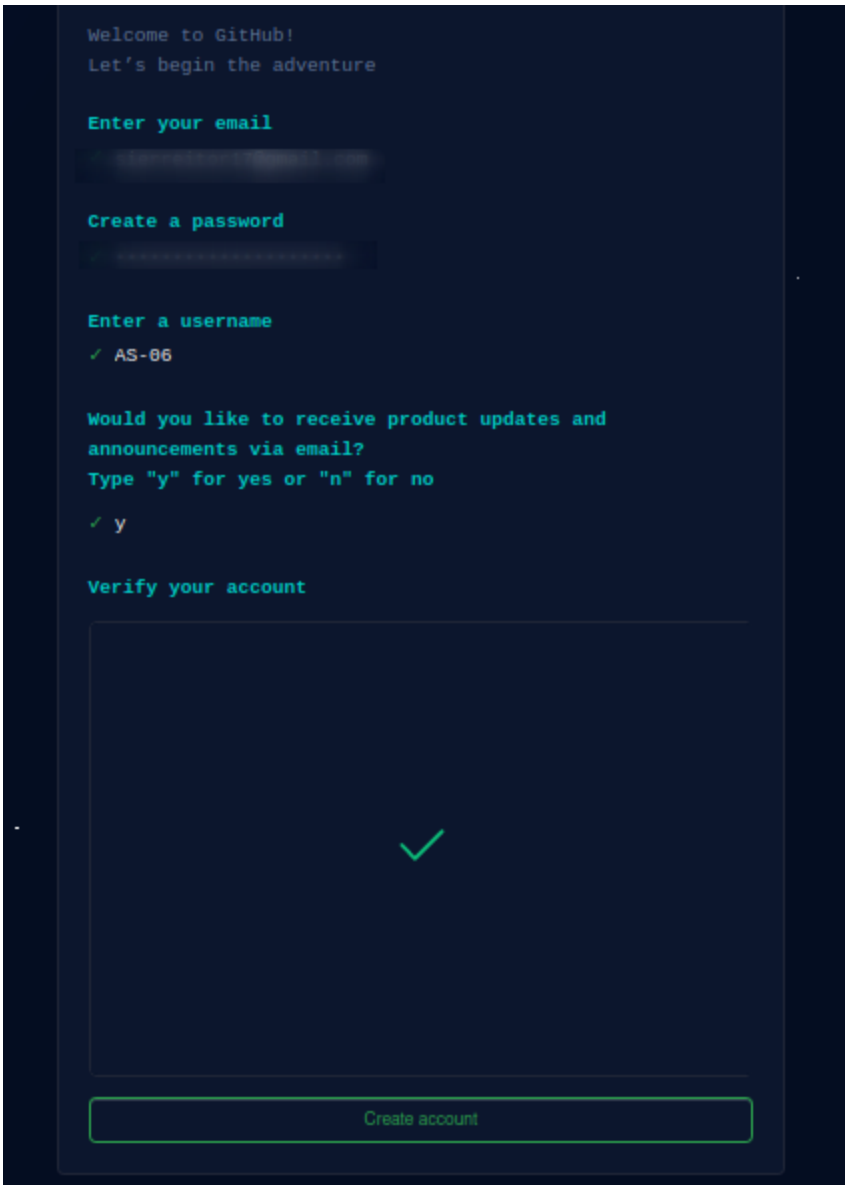
GIT-HUB

La creación de una cuenta en Git-Hub es muy sencilla solo es seguir los siguientes pasos y se realizara este proceso con éxito

1. Lo primero que se hará es ir a https://github.com/
2. Vamos a Sign up



3. Ingresamos los datos requeridos



4. Esperamos el código que llegara al correo ingresado para verificarlo
5. Después de verificar la cuenta seguimos los pasos dados por Git-Hub
6. Por ultimo configuramos el perfil y terminamos de crear la cuenta

Inicio de sesión por clave SSH

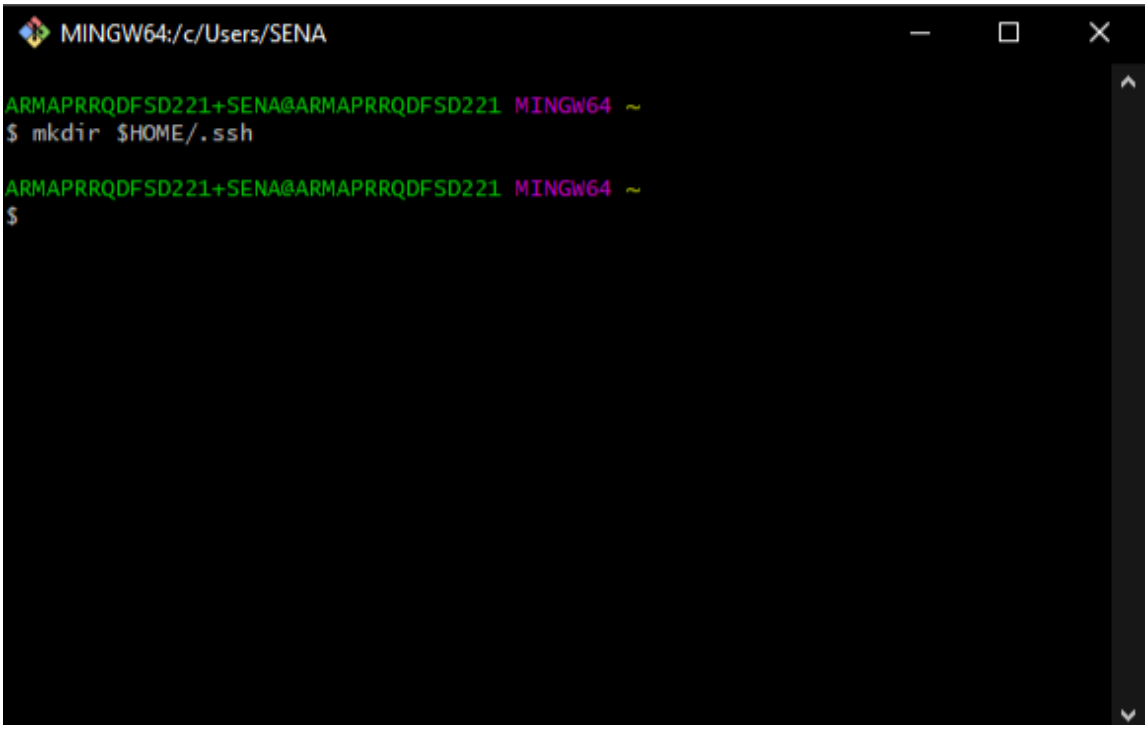
Una clave SSH le permite iniciar sesión en su servidor sin necesidad de una contraseña.

Esto se podrá usar tanto en Git Basch como en CMD, se **recomienda** trabajar en Git Bash.

A continuación podrá ver los pasos para generar la clave SSH:

1. Primer paso**, e**jecutar el siguiente comando para crear la carpeta “.ssh” en la raíz del usuario del ordenador:

   `mkdir $HOME/.ssh`



2. Luego se generan las claves con el siguiente comando:

   `ssh-keygen -t rsa -b 4096 -C tu@correo.com`

Después aparecerá el aviso de que las claves se han generado tanto la pública como la privada y mostrara en qué ubicación se va a guardar, este pedirá una contraseña de forma opcional, pero sino se desea ingresar la contraseña oprima la tecla enter dos veces.

3. Para verificar si las claves existen se realiza el siguiente comando:

   `ls -al ~/.ssh`

   Nota: Las claves SSH siempre se generan como un par de claves públicas (`id_rsa.pub`) y privadas (`id_rsa`). Es extremadamente importante **nunca revelar su clave privada**, y **sólo usar su clave pública** para cosas como la autenticación de GitHub.

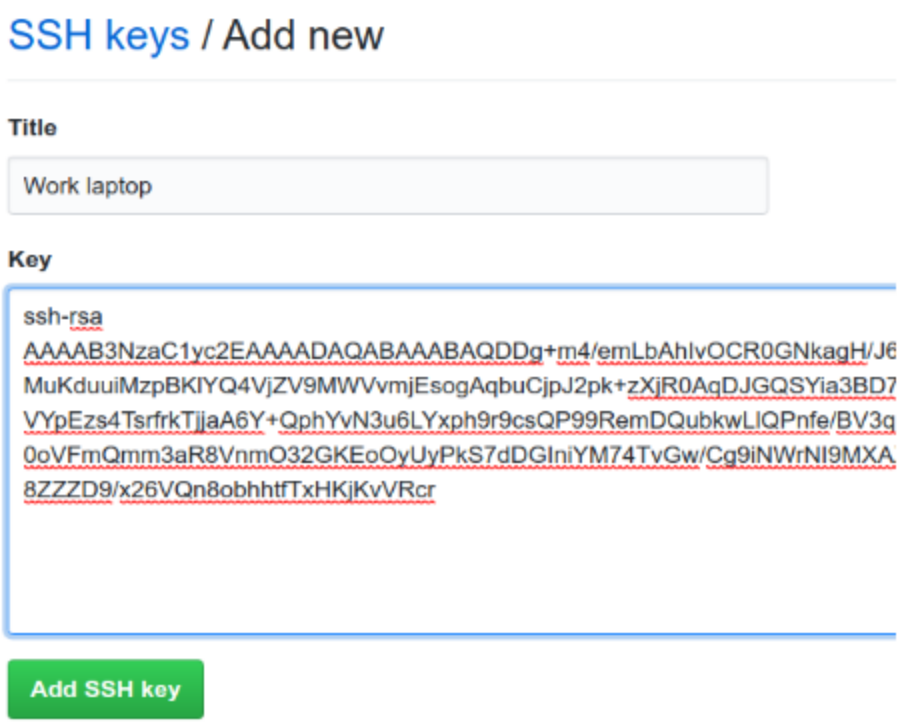
4. A continuación se copia la llave pública de SSH, para hacerlo se realiza de la siguiente forma:

   `clip < ~/.ssh/id_rsa.pub # Windows`

Agregar la clave SSH pública a GitHub

Ve a la página de configuración de tu GitHub y haz clic en el botón "New SSH key":

Luego dar a tu clave un título reconocible y pegarla en su clave pública (`id_rsa.pub`):



Finalmente, prueba la autenticación con:

ssh -T git@github.com

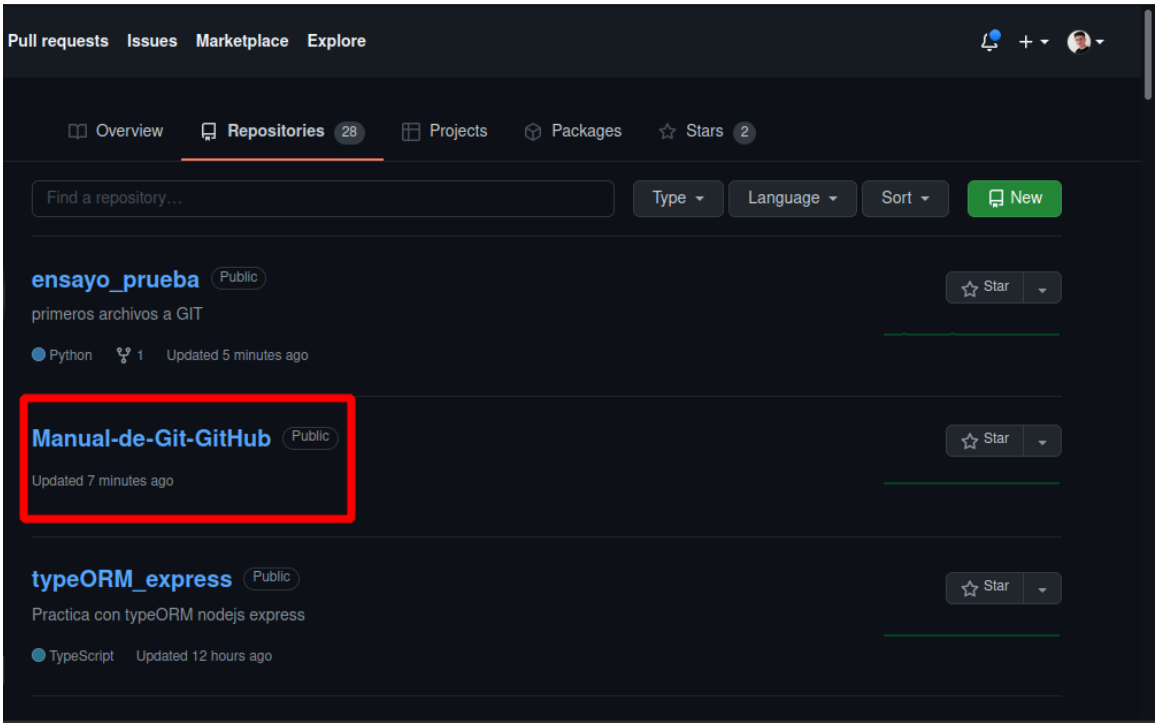
Si ha seguido todos estos pasos correctamente, deberías ver este mensaje:

Hi tu_usuario! You've successfully authenticated, but GitHub does not provide shell access.

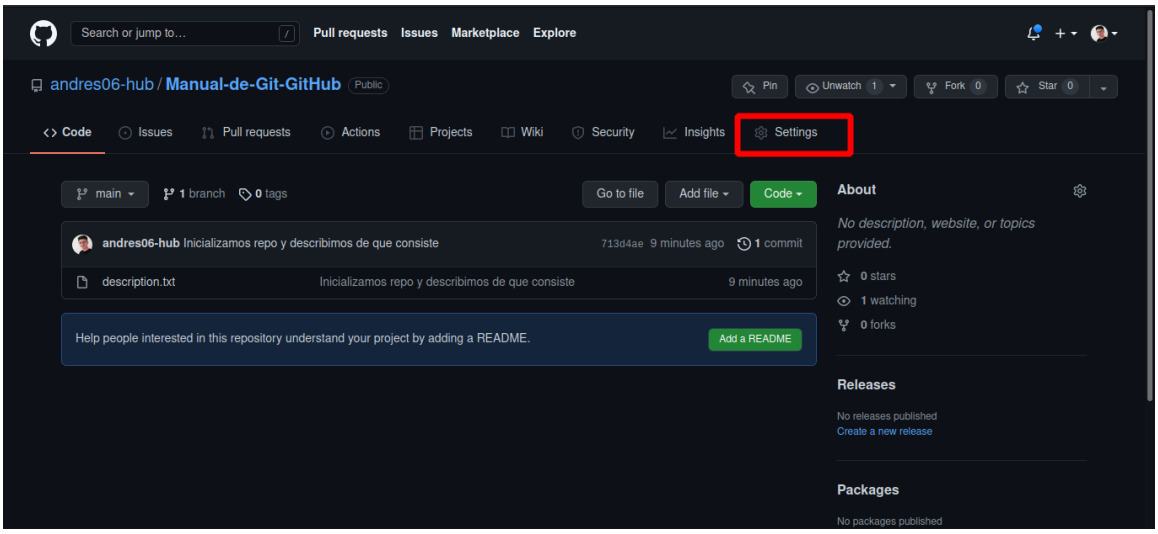
Invitar a otras personas a participar

Para invitar colaboradores a un proyecto de GitHub seguimos los siguientes pasos:

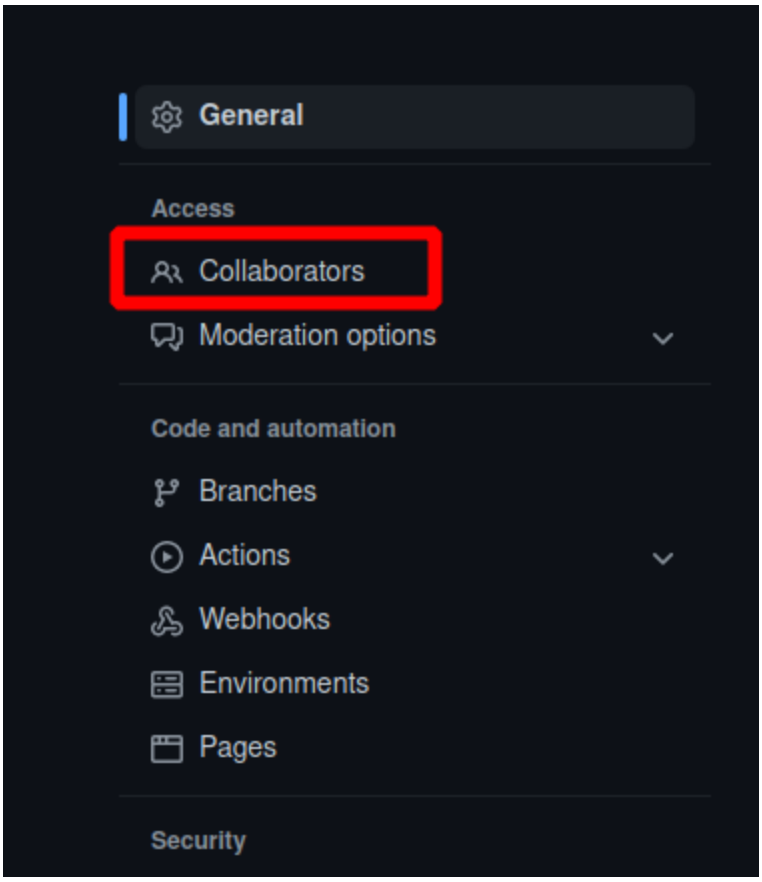
Creamos un repositorio en GitHub y entramos en el



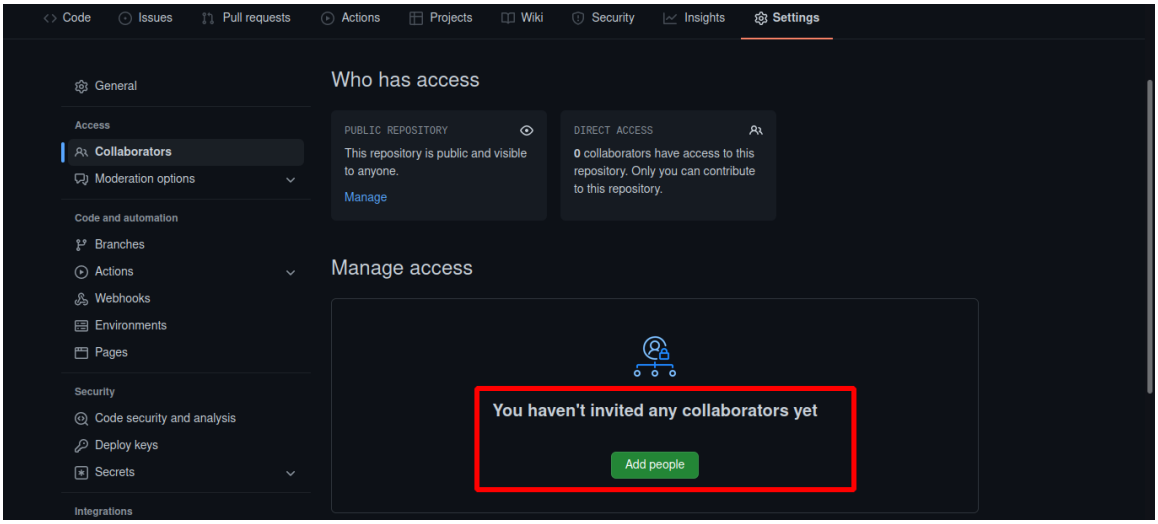
Al entrar en el repositorio creado vamos a configuraciones



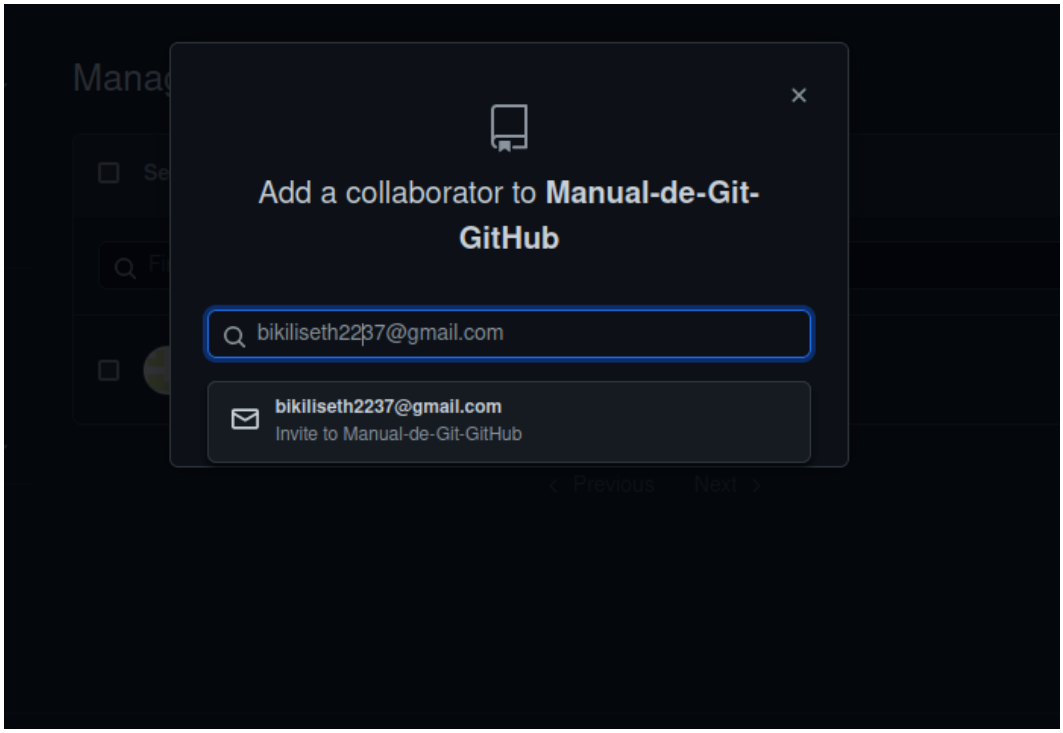
Después ingresamos a colaboradores

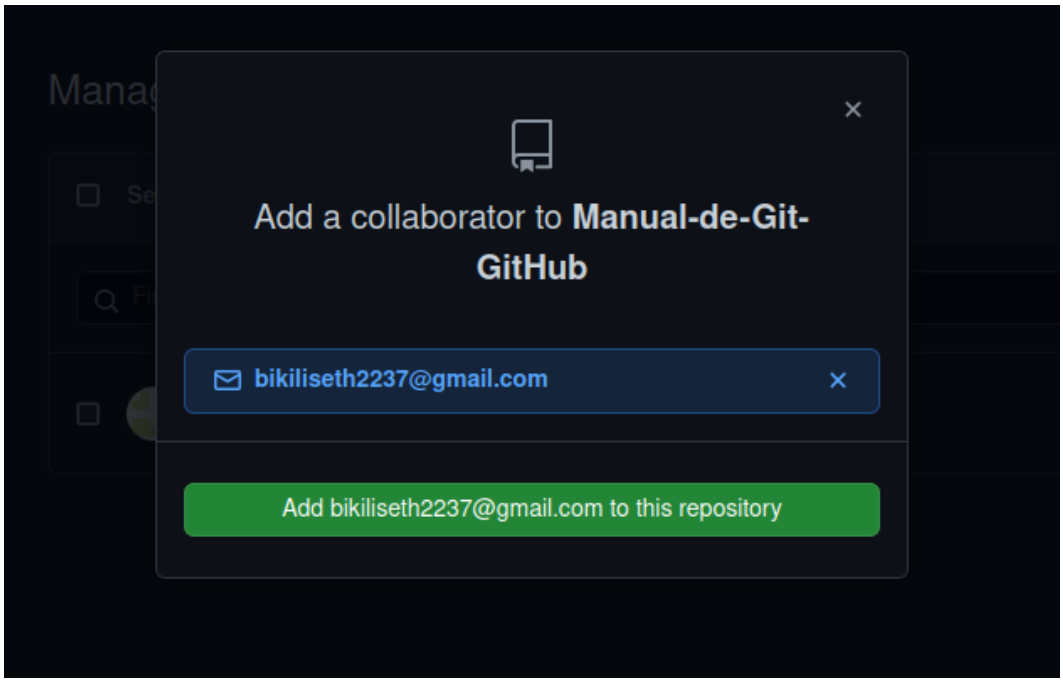


Vamos a la siguiente opción

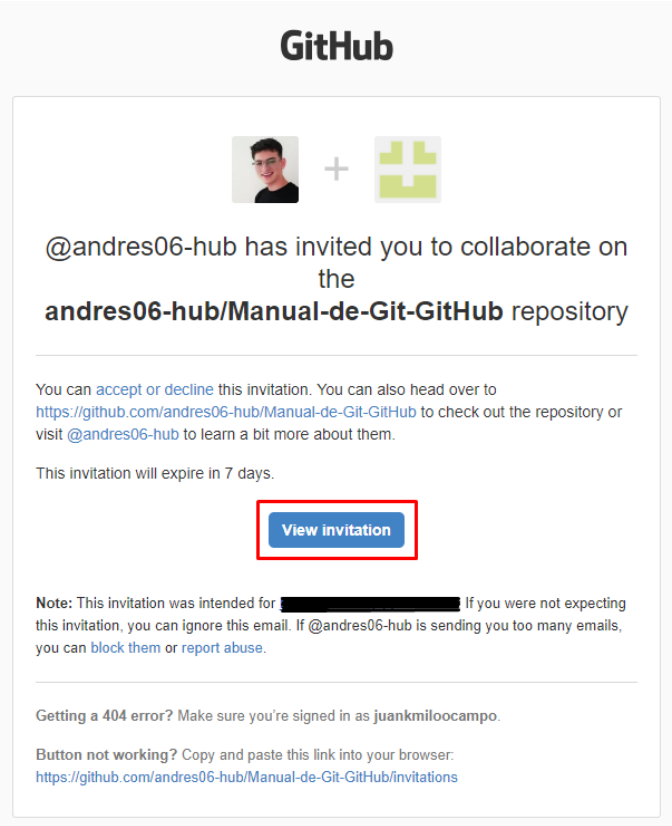


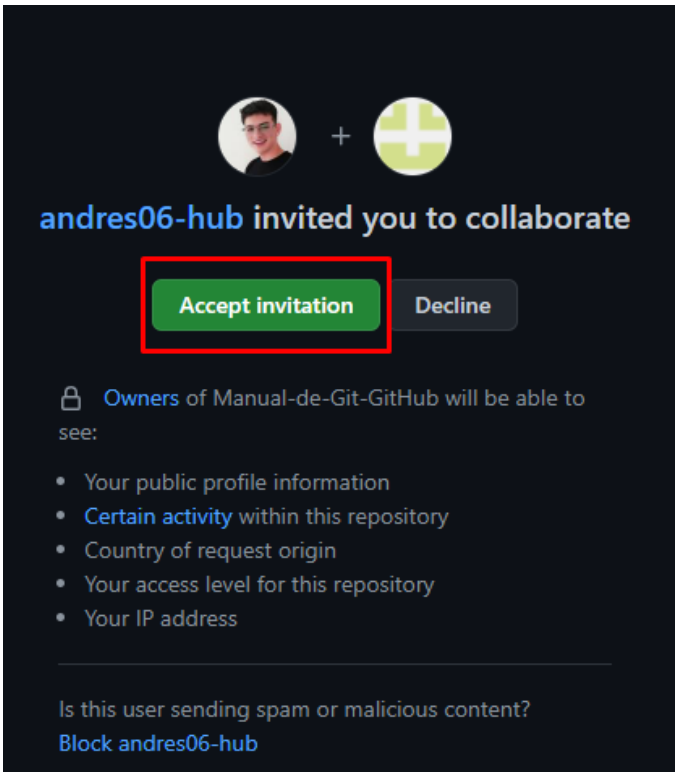
Para invitar gente se coloca el correo o usuario de Git-Hub de la persona



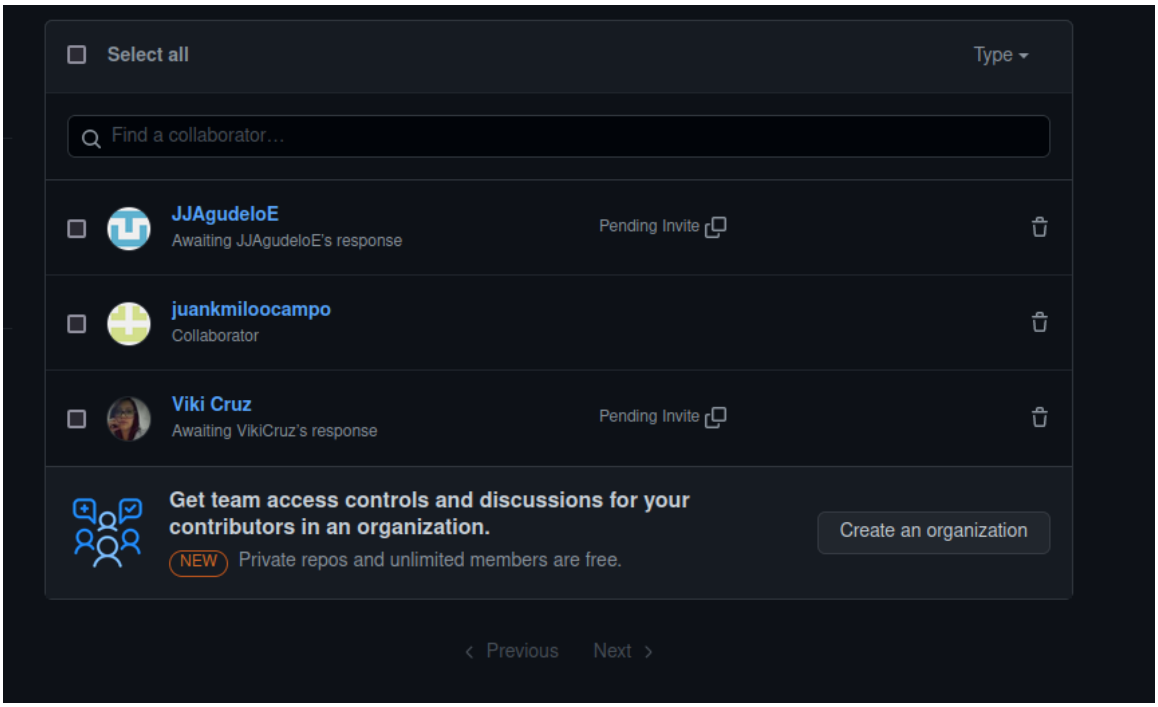


A los colaboradores les llegara una invitación al correo



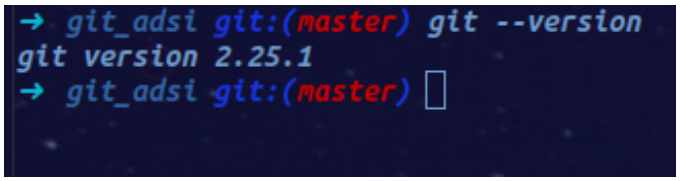


De esta manera se invitan integrantes a un repositorio



Comandos Git & GitHub

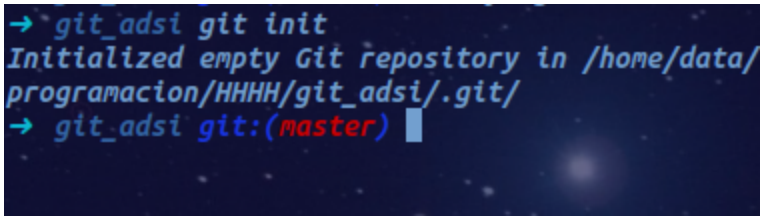
- Para saber la versión de ‘git’ que tenemos, ejecutamos el siguiente comando



- Si queremos crear un repositorio local primero creamos una carpeta.

con el comando `mkdir` creamos la carpeta seguido del nombre de ella EJ: `mkdir <nombre_carpeta>`

- Así mismo ingresamos a la carpeta creada e iniciamos un repositorio local, con el siguiente comando `git init`



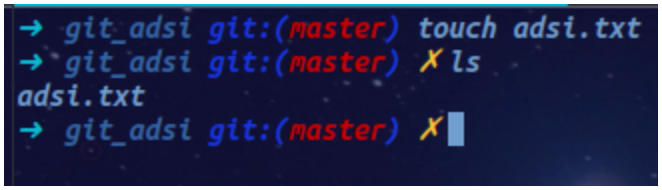
→Si vemos la carpeta nos mostrara el git inicializado con `git:(master)`

- Si queremos eliminar el `git` iniciado en la carpeta, ejecutamos el siguiente comando

  `rm -rf .git`

- Creamos un archivo en nuestra carpeta para poder subirlo o guardarlo al repositorio creado(local)

  Linux → `touch <nombre_archivo>`



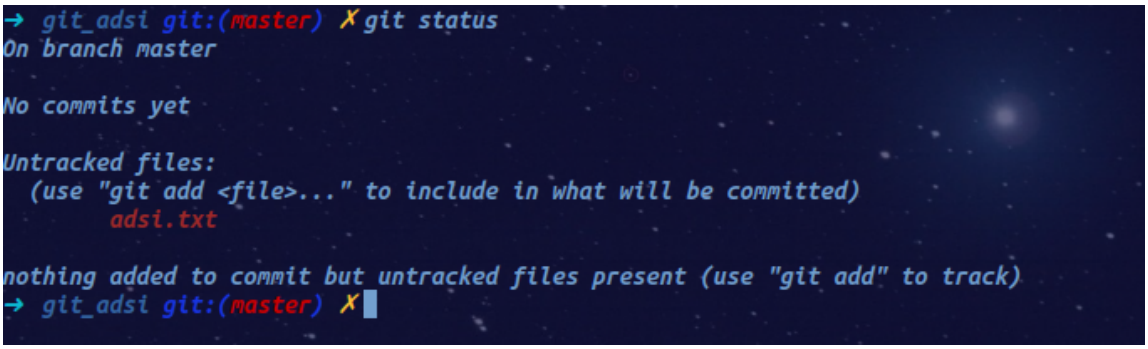
  Windows → `COPY CON <nombre_archivo>`

```
C:\Users\carlos\pruebas>COPY CON archivo1.txt
Texto de prueba.
^Z
        1 archivo(s) copiado(s).

C:\Users\carlos\pruebas>DIR
 El volumen de la unidad C no tiene etiqueta.
 El número de serie del volumen es: 1084-9497

 Directorio de C:\Users\carlos\pruebas

05/04/2016  16:34    <DIR>          .
05/04/2016  16:34    <DIR>          ..
05/04/2016  16:34                18 archivo1.txt
05/04/2016  00:31    <DIR>          d1
05/04/2016  00:31    <DIR>          d2
               1 archivos             18 bytes
               4 dirs  473.330.077.696 bytes libres

C:\Users\carlos\pruebas>
```

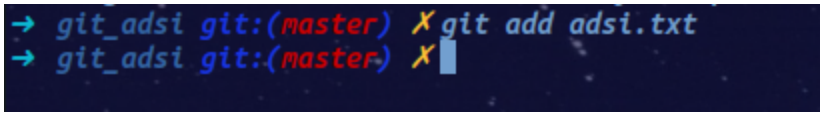
Después de crear el archivo entramos a él y editamos su contenido

- Cuando hay un movimiento o si editamos algo del repositorio y queremos saber si este tuvo cambios, ejecutamos este comando `git status` (Recordar que los comandos se ejecutan en donde esta inicializado el repositorio)

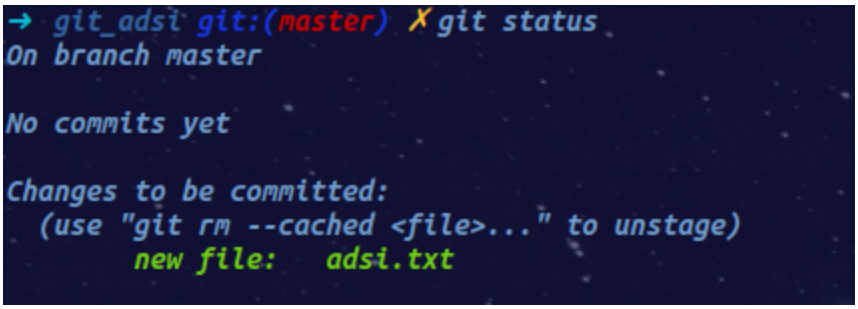
  →Este nos da el estado del repositorio



- Para agregar o marcar todos los archivos modificados , en este caso `adsi.txt` informa que tuvo un cambio, entonces lo guardamos ,marcamos o agregamos con el siguiente comando `git add <nombre_archivo>` (archivo en especifico) ó `git add .` (para guardar todos los cambios del repositorio local)



→ Podemos ver los datos que esta almacenado temporalmente con el `git status` que nos aparecerá los archivos guardados en verde u otro color

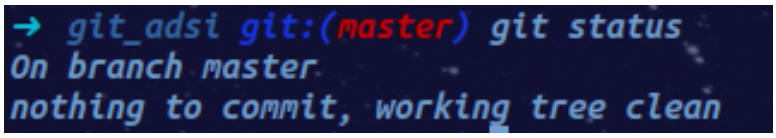


TODO

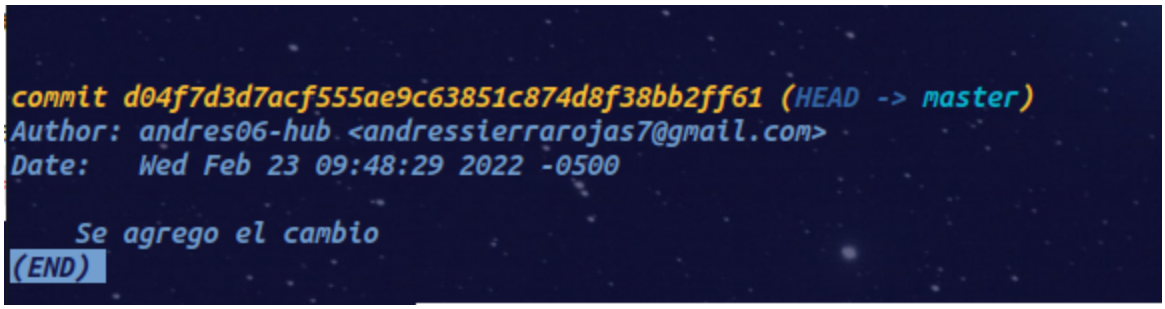
- Para almacenar los datos y pasar de `index` a un repositorio definitivo (`head`), el comando seria `git commit`

El `git commit` acepta comentarios para así darle una descripción o comentario del movimiento. Ejemplo : `git commit -m “<mensaje>”`

- Se puede volver a revisar el estado del repositorio con `git status`



- También es posible revisar el historial de cambios o movimientos con `git log`



Este nos muestra :

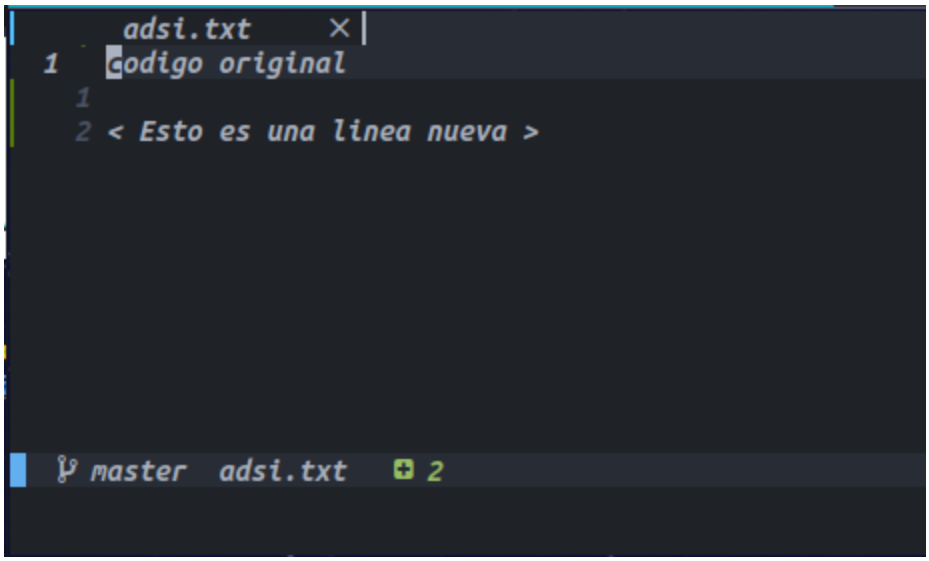
→El hash del commit

→ El autor del repositorio

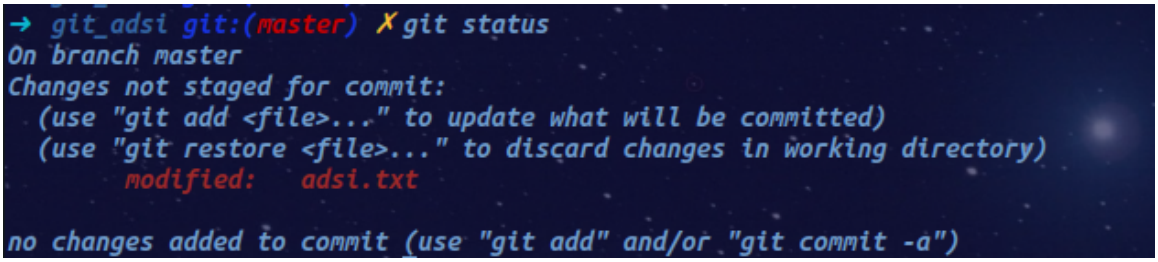
→ La fecha y hora que se hizo

→Nos muestra el mensaje puesto en el commit

A continuación veremos el manejo de las versiones con `git`, cuando se edita un archivo ya existente.



Se obtendrá el estado del repositorio, se puede observar que `git` nos informa el cambio realizado : `git status` (Para saber el estado del repositorio)



→Nos aparece que el archivo `adsi.txt` fue modificado

- Si desea que los cambios no se guarden o sean descartados, se ejecuta `git checkout`

- Por otro lado si desea guardar los cambios realizados ejecute el comando mencionado anteriormente `git add <nombre_archivo>` o `git add .` (Para guardar todos los archivos modificados)

```
➜ git_adsi git:(master) ✗ git add .
➜ git_adsi git:(master) ✗
```

- Luego de esto ejecute el `git commit -m “<mensaje>”`

```
➜ git_adsi git:(master) ✗ git commit -m "Se edito el archivo"
[master cf37cfd] Se edito el archivo
 1 file changed, 2 insertions(+)
```

- Por ultimo se revisa el historial de cambios con `git log` , aparecerá los dos movimientos que se han realizado

```
commit cf37cfd63456d362f37130f1a0b49d9dd06221a7 (HEAD -> master)
Author: andres06-hub <andressierrarojas7@gmail.com>
Date:   Wed Feb 23 11:09:46 2022 -0500

    Se edito el archivo

commit d04f7d3d7acf555ae9c63851c874d8f38bb2ff61
Author: andres06-hub <andressierrarojas7@gmail.com>
Date:   Wed Feb 23 09:48:29 2022 -0500

    Se agrego el cambio
(END)
```

- Para regresar a una versión anterior de los archivos almacenados en **git** se usa el comando

`git checkout <hash>` ó `git checkout <primero_7_digitos_hash>`

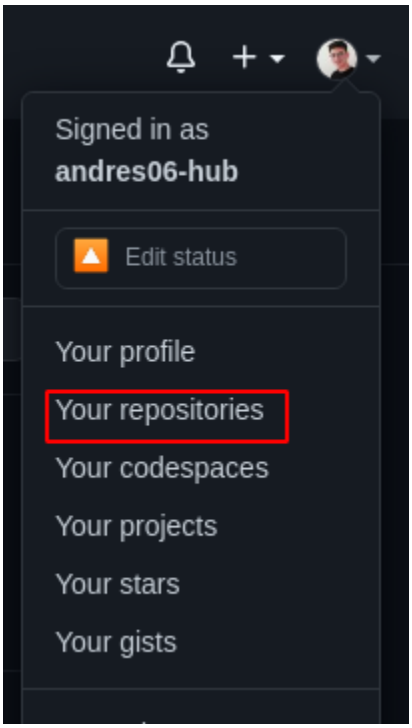
EJ: `git checkout cf37cfd`

Y el comando `git checkout <nombre_rama>` se usa para regresar a su ultima versión almacenada

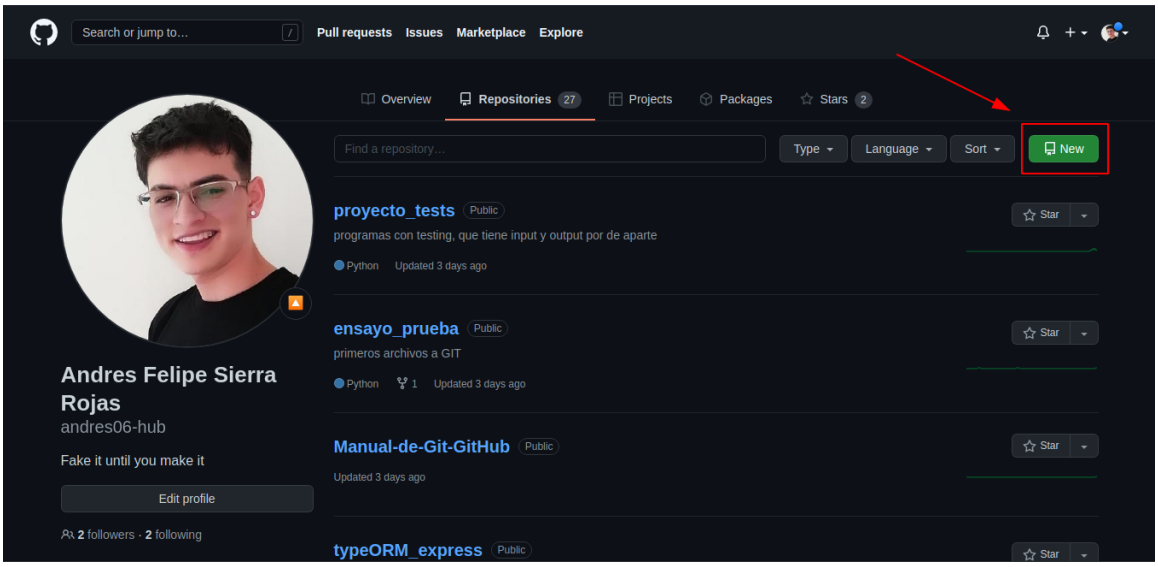
Ejemplo: `git checkout master`

Repositorios Remotos

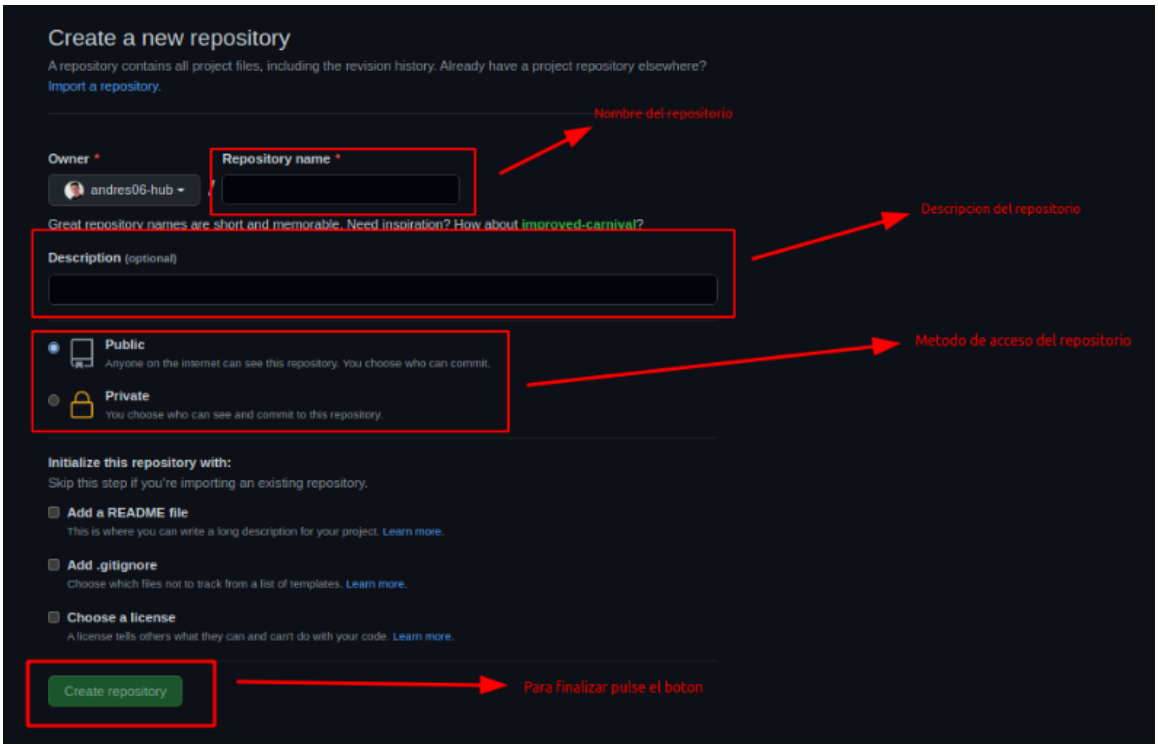
Para trabajar con repositorios remotos (en este caso se trabajara con GitHub pero también se puede con GitLab), se crea una cuenta en GitHub y se ingresan los datos solicitados, a continuación ingrese a **your repositories** donde se pueden ver alojados los repositorios creados.



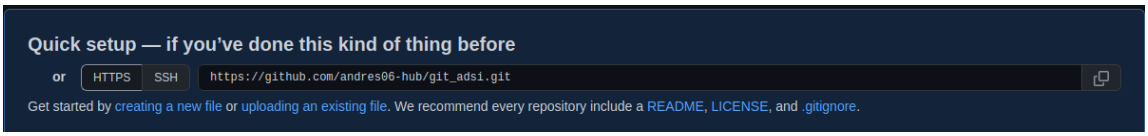
La creación de un nuevo repositorio seria ingresando a:



se crea el repositorio en GitHub ingresando el nombre que se le asignara y se configura su metodo de acceso ya sea publico o privado



- Después de crearlo aparecerá una guía de como subir o enlazar repositorios y en la parte superior muestra las opciones para enlazarlo



en este caso se hará por **https pero tambien se podria hacer por SSH.**

En caso de no tener un repositorio local se pueden seguir los pasos anteriores. Ejemplo:

→Se crea la carpeta donde estará el repositorio `mkdir <nombre_carpeta>`

→Se ingresa a la carpeta creada y se inicializa el repositorio `git init`

→Luego se crean los archivos que estarán en el repositorio

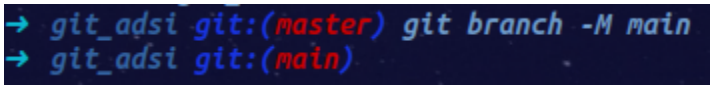
→Despues se consulta el estado del repositorio con `git status` y para agregarlo al index se usa el comando `git add .`

→Se realiza un commit con `git commit -m "<mensaje o descripcion>"` para asi guardarlo en el 'head'

- Se observa en que rama esta el repositorio, se sabe que GitHub no trabaja con la rama `master` sino con la `main`

  En este caso el repositorio esta en la `master`

para cambiar la rama solo es ejecutar el siguiente comando



decimos que de la rama `-M` se pase a la `main`

- Aquí se crea el enlace de los repositorios, tanto el local como el remoto

  → Ejecutamos el comando `git remote add <alias del remoto> <SSH ó https>`

```
➜ git_adsi git:(main) git remote add origin https://github.com/andres06-hub/git
_adsi.git
➜ git_adsi git:(main)
```

Para saber con que alias quedo nuestro repositorio se ejecuta `git remote`

```
➜ git_adsi git:(main) git remote
origin
➜ git_adsi git:(main)
```

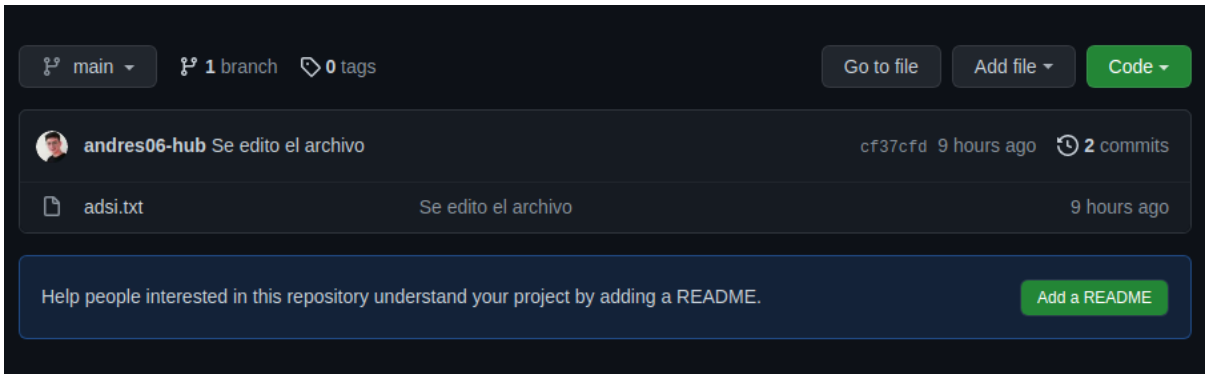
y se verifica en que estado se encuentra el repositorio local `git status`

```
➜ git_adsi git:(main) git status
On branch main
nothing to commit, working tree clean
➜ git_adsi git:(main)
```

Acá es donde se envia o se carga el repositorio local al remoto con `git push -u <nombre_remoto> <rama>` en este caso seria `git push -u origin main`

```
➜ git_adsi git:(main) git push -u origin main
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (6/6), 503 bytes | 503.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0)
To https://github.com/andres06-hub/git_adsi.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
➜ git_adsi git:(main)
```

Y se puede ver en el GitHub el cambio que tendrá el repositorio o tambien se pueden observar los `commits` y los cambios de los repositorios

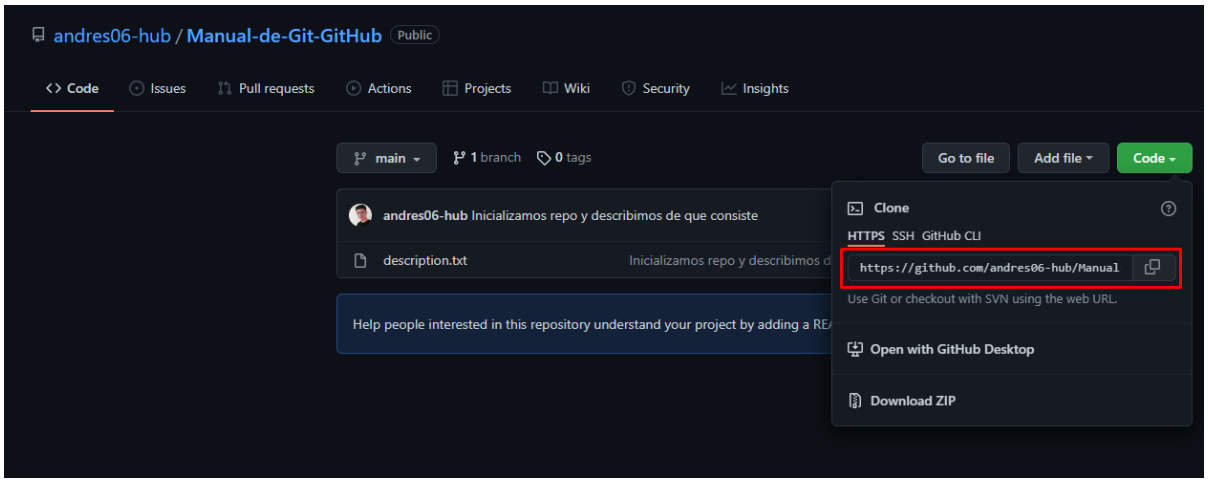


Git clone

Cuando existe un repositorio creado en GitHub y se desea clonar en el repositorio local se usa el siguiente comando `git clone <http ó ssh>` ubicados en la carpeta donde se desea guardar.

Ejemplo:

Se copia el `https` ó `ssh` del repositorio remoto para hacer el git clone



Y de esta manera se clona el repositorio

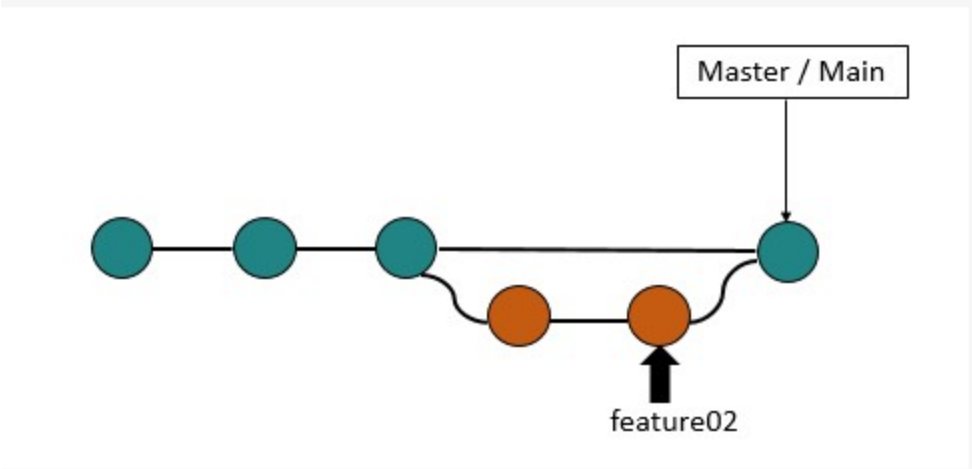
```
C:\Users\SENA\Desktop\Manejo de ramas>git clone https://github.com/andres06-hub/Manual-de-Git-GitHub.git
Cloning into 'Manual-de-Git-GitHub'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.

C:\Users\SENA\Desktop\Manejo de ramas>
```

Manejo de ramas

Para tener un control de versiones en una rama diferente a la `master` se crean nuevas ramas para que cada integrante trabaje en una rama única, evitando afectar la parte principal del proyecto

→Para poder crear ramas en un repositorio se ejecuta el siguiente comando `git branch <nombre-de-la-rama>`

Un claro ejemplo de como se podría ver seria



Como se puede observar en la imagen se tiene una rama principal (`main:master`) donde se encuentra el repositorio y en la tercera esfera se usa el comando `git branch <nombre-de-la-rama>` para crear una nueva rama en donde se pueden llevar otros desarrollos sin afectar la `main`

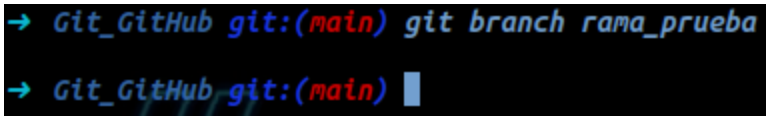
Para ver las ramas creadas se ejecuta el siguiente comando `git branch --list` (recordar que para hacer esto, se tienen que manejar los temas explicados anteriormente y no obstante, tener un repositorio inicializado)

1. Ver ramas creadas

   `git branch /ó/ git branch --list`

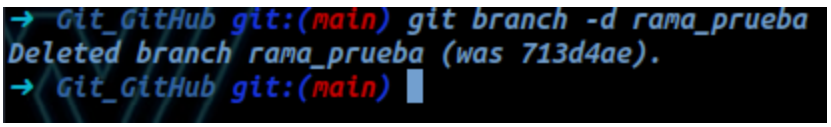
   → En este caso existe únicamente la rama principal (`main`)

2. Para crear una rama se usa el comando `git branch <nombre_rama>`



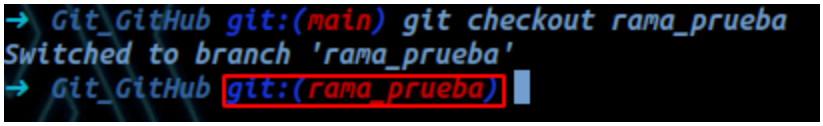
   Y se puede observar la nueva rama creada (`rama_prueba`) ingresando nuevamente el comando `git branch --list`

3. EL comando `git branch -d <nombre_rama>` se usa para eliminar una rama:



   Se puede observar nuevamente la lista de las ramas con el comando `git branch --list` para confirmar que ya no existe

4. El comando `git checkout <nombre_rama>` sirve para ubicarse en otra rama:

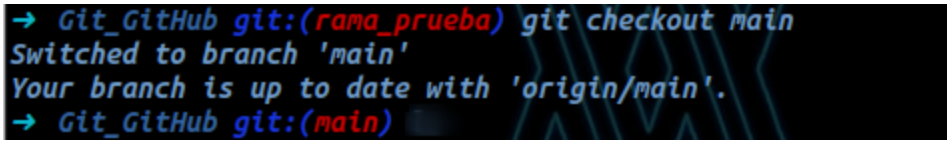


5. Para subir una rama al repositorio remoto se usa `git push -u origin <nombre-de-rama>`

Fusionar Ramas

Se pueden unir las ramas creadas en el proyecto, fusionando los cambios a la rama principal, esto se puede realizar con el comando: `git merge <rama_a_unir>`

→Primero se debe ubicar en la rama principal `main - merge` con el comando `git checkout main`



→Para unir las dos ramas use `git merge <rama_a_unir>`