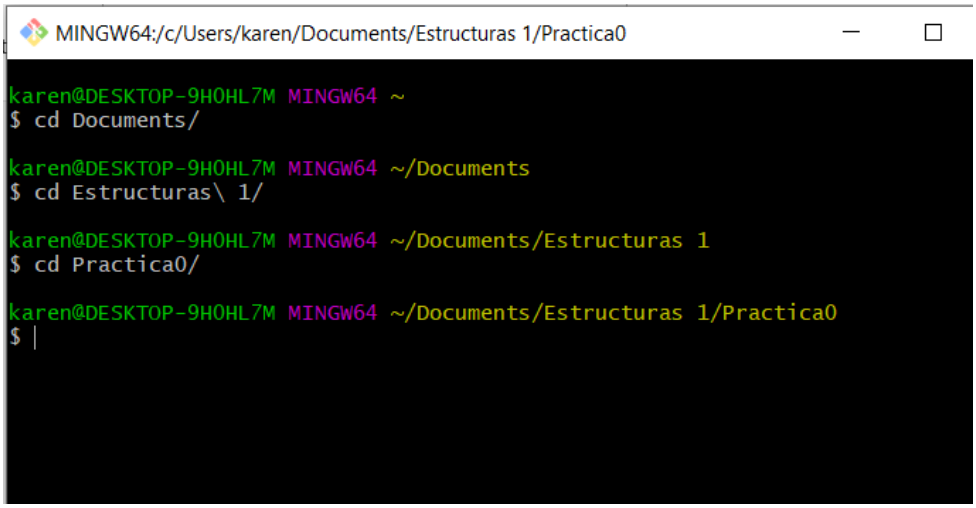
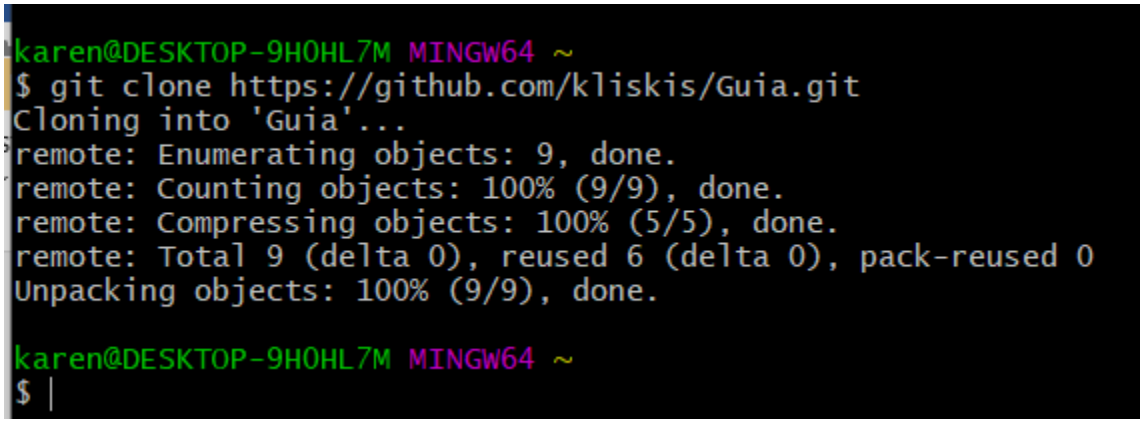
Guía GIT

Utilizando el comando "cd" navegar hasta donde se tiene el proyecto:

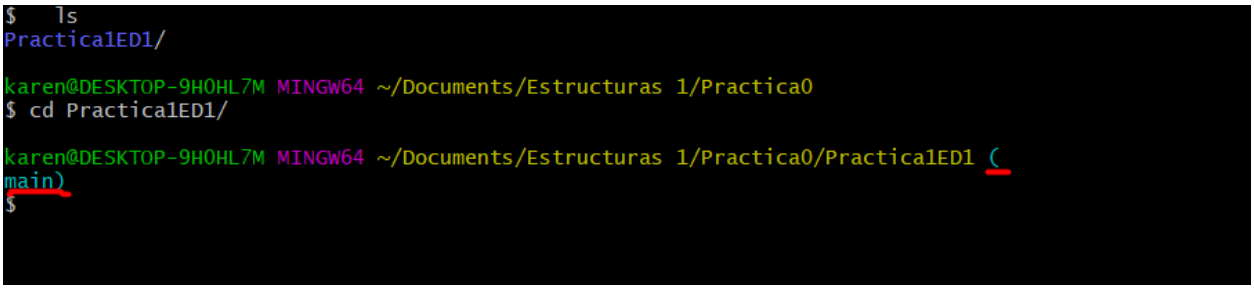


Clonar el proyecto creado en github utiliznado el comando git clone <URL proporcionada por Github>



Entrar en la carpeta del proyecto que se ha clonado, donde se encuentra la carpeta escondida .git

Una vez allí, notará un agregado en el camino mostrado en consola, esta es la rama actual, por defecto será "main" o "master"



Para Verificar si existen cambios sobre los archivos existentes, deberá ejecutar el comando git status, el cual, mostrará el detalle de los cambios realizados:

```
karen@DESKTOP-9HOHL7M MINGW64 ~/Documents/Estructuras 1/Practica0/Practica1ED1 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")

karen@DESKTOP-9HOHL7M MINGW64 ~/Documents/Estructuras 1/Practica0/Practica1ED1 (main)
$
```

Para agregar los cambios, deberá ejecutar el comando "git commit" para agregar cambios, encapsular cambios, describir cambios, esto llega al **Repositorio Local.**

Lo recomendado: acompañar git commit con los parámetros:

- -a : significa que incluirá todos los cambios presentes
- -m : para incluir el mensaje, que a su vez acepta el mensaje entre comillas.

De lo contrario abrirá un espacio para agregar el mensaje.

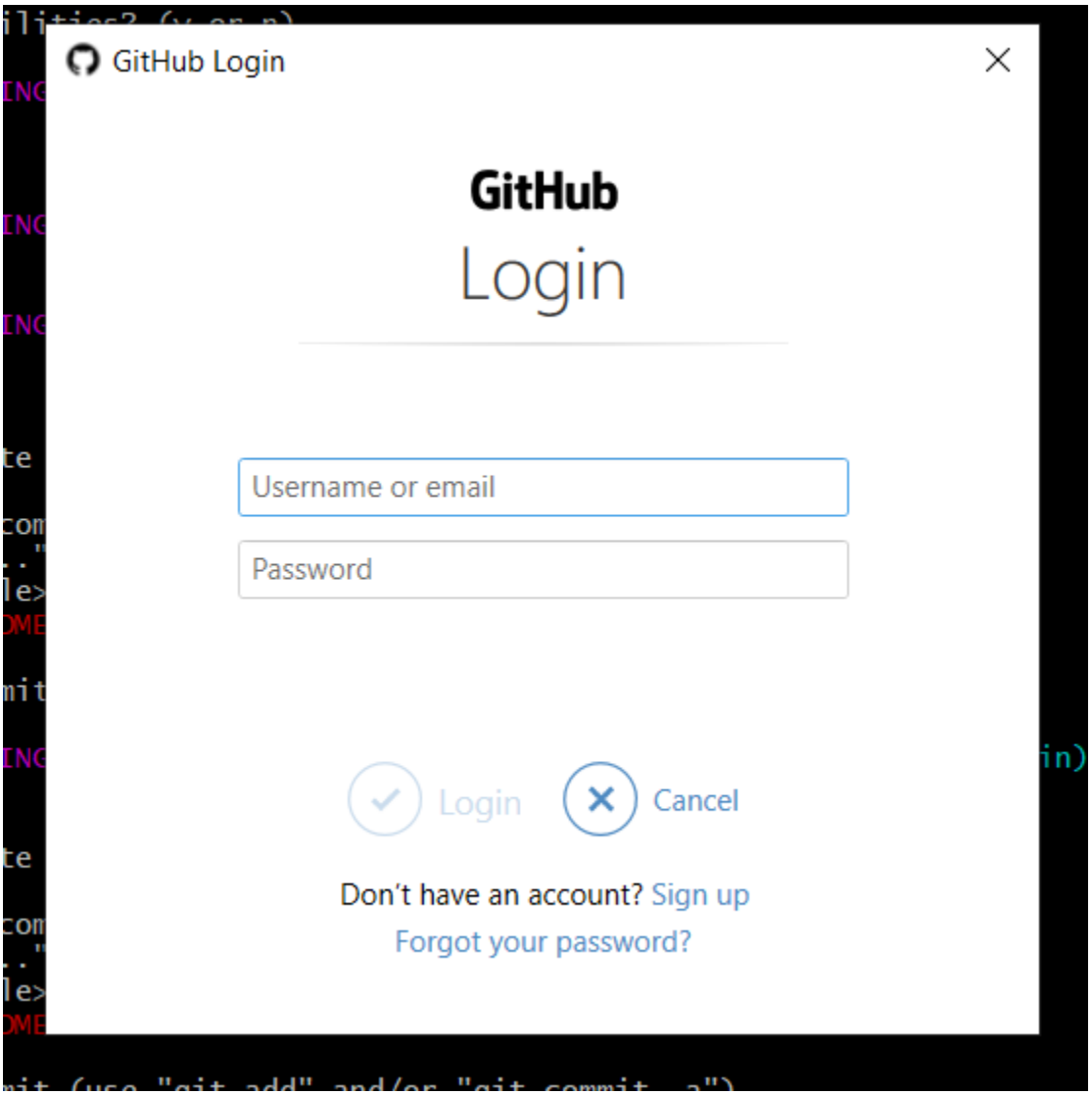
```
karen@DESKTOP-9HOHL7M MINGW64 ~/Documents/Estructuras 1/Practica0/Practica1ED1 (main)
$ git commit -a -m "Primer commit"
[main 0a41c14] Primer commit
 1 file changed, 4 insertions(+), 1 deletion(-)

karen@DESKTOP-9HOHL7M MINGW64 ~/Documents/Estructuras 1/Practica0/Practica1ED1 (main)
$
```

Para publicar los cambios realizados deberá ejecutar el comando "git push":

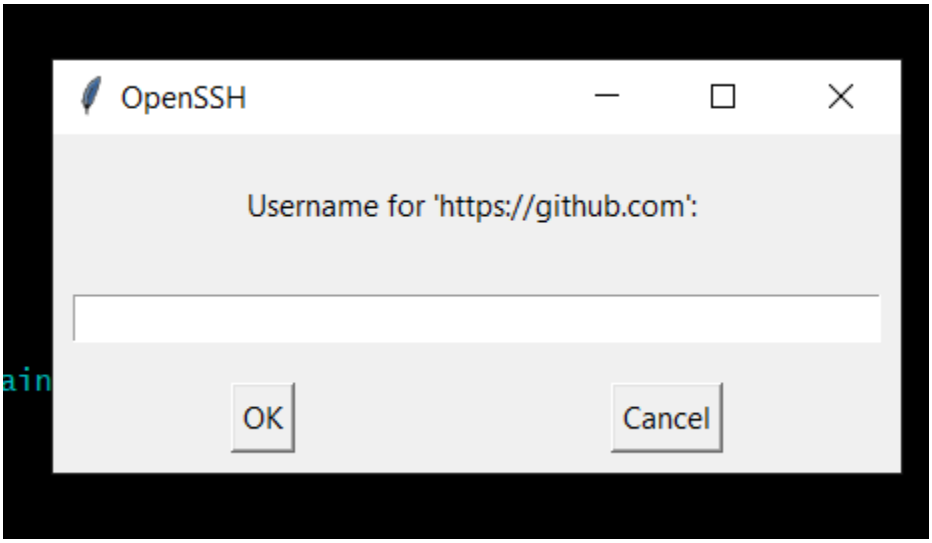
```
karen@DESKTOP-9HOHL7M MINGW64 ~/Documents/Estructuras 1/Practica0/Practica1ED1 (main)
$ git push
```

Si es la primera vez que se ejecuta esta acción, se solicitarán credenciales:

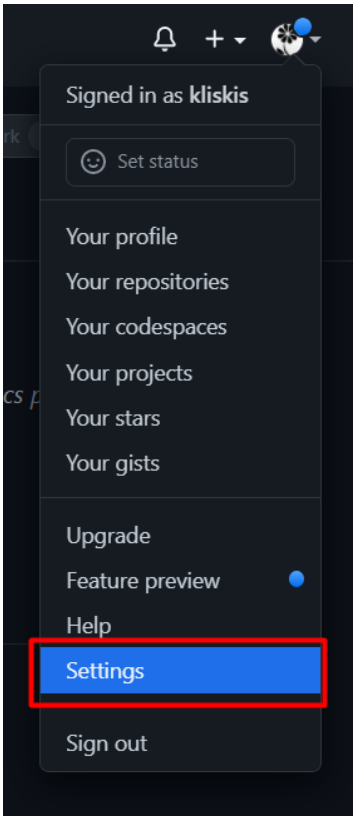


Luego de ingresar el usuario y la contraseña, se pedirá algo más por motivos de seguridad:

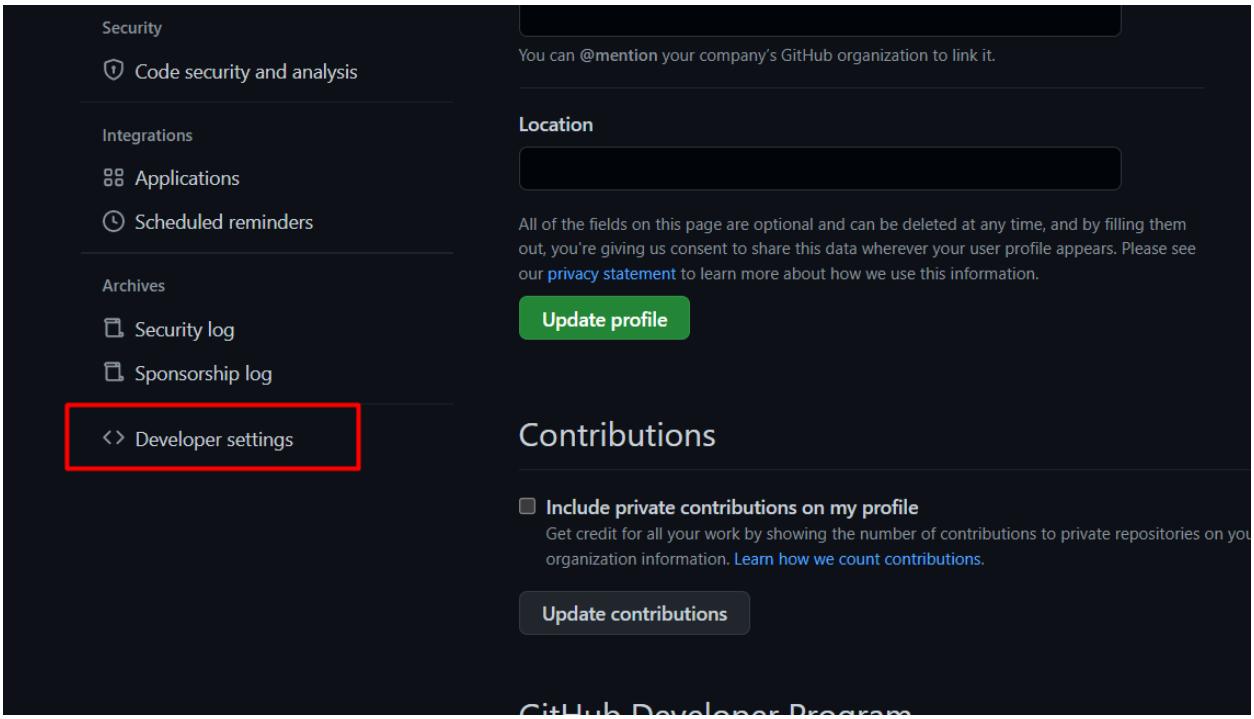
```
karen@DESKTOP-9H0HL7M MINGW64 ~/Documents/Estructuras 1/Practica0/Practica1ED1 (main)
$ git push
Logon failed, use ctrl+c to cancel basic credential prompt.
```



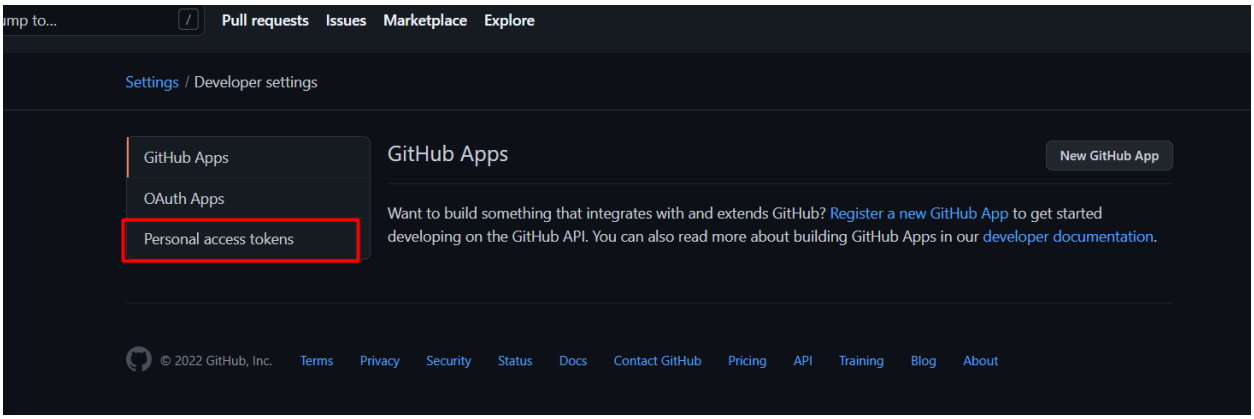
Para cumplir todos los requisitos de autenticación, deberá crear un "Access token"



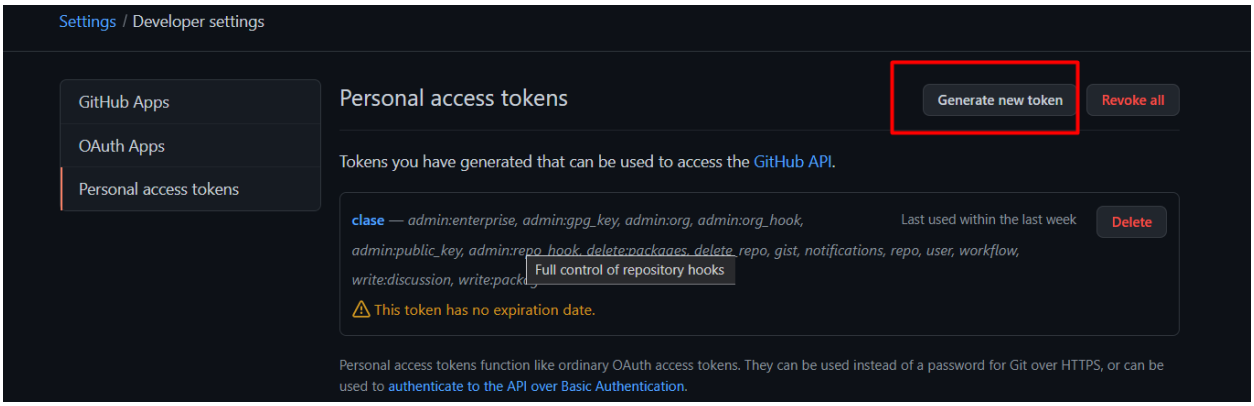
Buscar los "developer settings" en el menú de la izquierda:



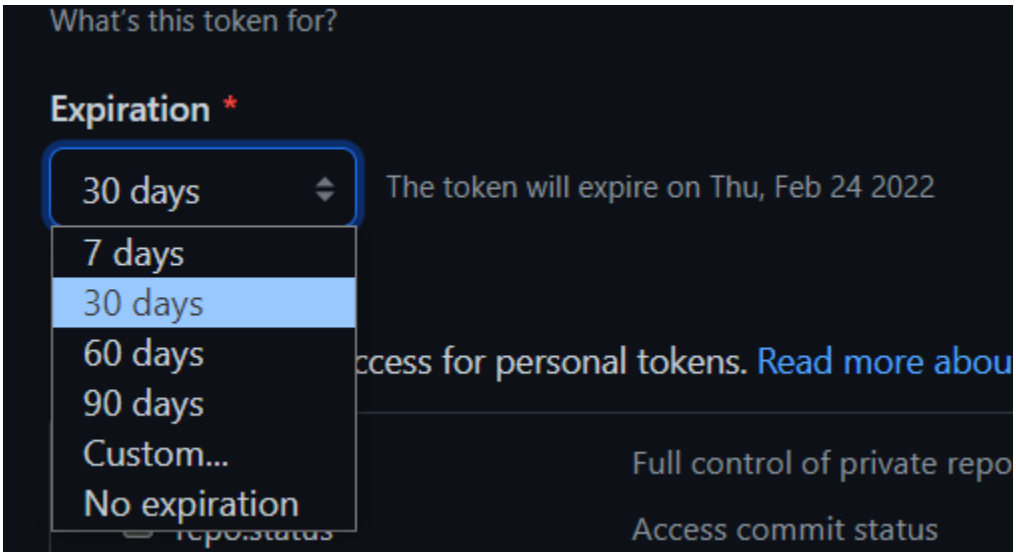
Seleccionar la siguiente opción:



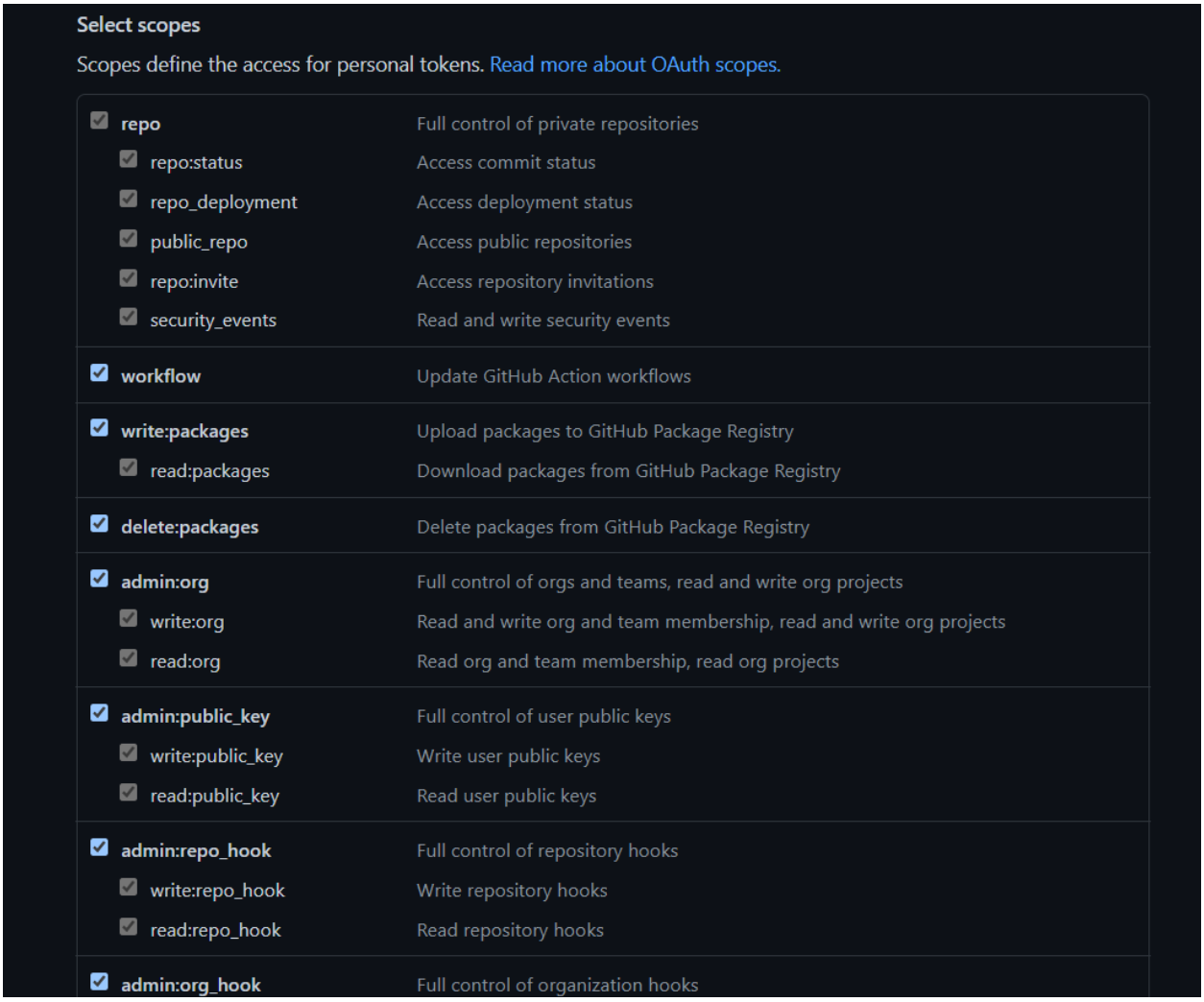
Generar un nuevo "Access Token":



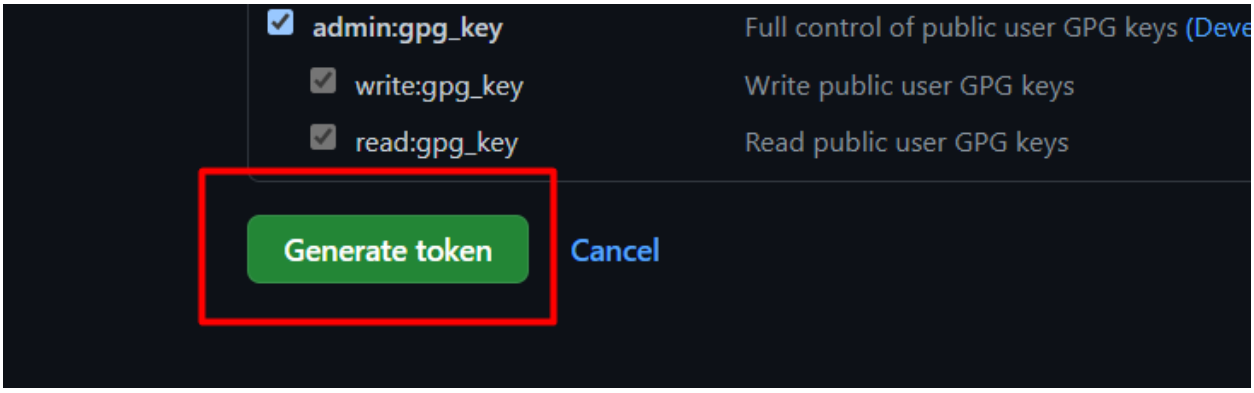
Escoger la expiración, recordando que debe renovarse una vez expire:



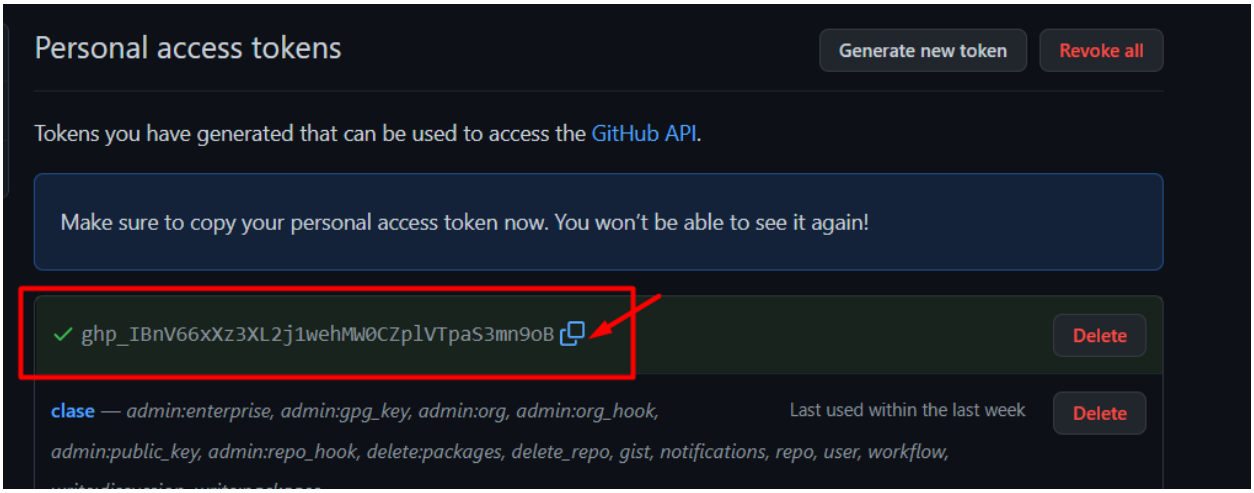
Lo recomendable es asignar todos los permisos para poder tener control total sobre el repositorio:



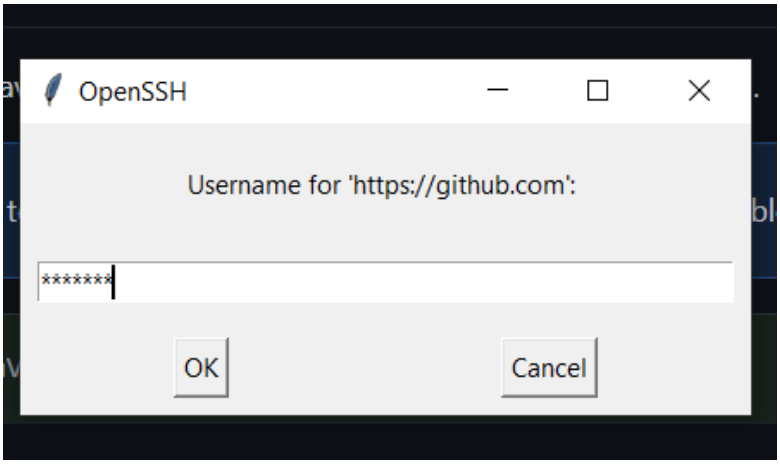
Finalmente se generar el token:



Deberá copiarlo y guardarlo o recordarlo como parte de las credenciales de la PC ya que no podrá verlo nuevamente por motivos de seguridad:



Ingresar el usuario



Pegar el token generado como contraseña en la siguiente opción:

Una vez realizado esto, el push se ha completado, lo que significa que los cambios están ahora en el **repositorio Remoto:**

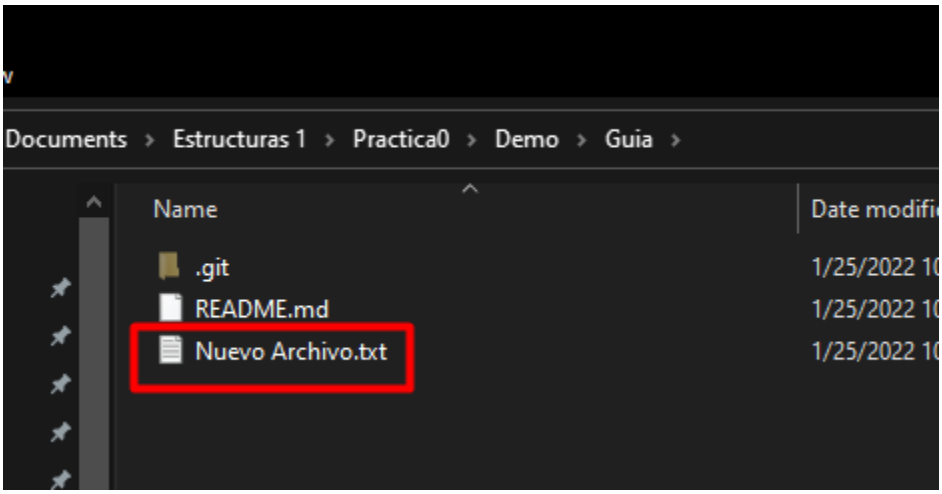
```
karen@DESKTOP-9H0HL7M MINGW64 ~/Documents/Estructuras 1/Practica0/Practica1ED1 (main)
$ git push
Logon failed, use ctrl+c to cancel basic credential prompt.
error: unable to read askpass response from 'C:/Program Files/Git/mingw64/libexec/git-core/git-gui--askpass'
Username for 'https://github.com': kliskis
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 304 bytes | 304.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/kliskis/Practica1ED1.git
   95f51de..0a41c14  main -> main

karen@DESKTOP-9H0HL7M MINGW64 ~/Documents/Estructuras 1/Practica0/Practica1ED1 (main)
```

Si revisamos el repositorio Remoto desde la página de Github:

Los cambios se han grabado y ahora se reflejan en el sitio.

Si incluimos nuevos archivos o carpetas, el comando “git Add -A” es el indicado para agregarlos a la solución:



```
karen@DESKTOP-9H0HL7M MINGW64 ~/Documents/Estructuras 1/Practica0/Demo/guia (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Nuevo Archivo.txt

nothing added to commit but untracked files present (use "git add" to track)
```

Git Add -A (agregar todo)

```
karen@DESKTOP-9H0HL7M MINGW64 ~/Documents/Estructuras 1/Practica0/Demo/guia (main)
$ git add -A
```

Nuevamente, para que los cambios queden almacenados como histórico deberá proceder con un commit:

```
karen@DESKTOP-9H0HL7M MINGW64 ~/Documents/Estructuras 1/Practica0/Demo/guia (main)
$ Git commit -a -m "Agregando archivo"
[main 4681b3a] Agregando archivo
 1 file changed, 1 insertion(+)
 create mode 100644 Nuevo Archivo.txt
```

Y para publicarlos con un git push:

```
karen@DESKTOP-9H0HL7M MINGW64 ~/Documents/Estructuras 1/Practica0/Demo/guia (main)
$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 291 bytes | 291.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/kliskis/Guia.git
   a1a52a6..4681b3a  main -> main
```

Branches:

La rama por defecto es la Master o Main, en ella, usualmente se guardan los cambios finales de un proyecto. Pero para trabajar distintos cambios, o requerimientos se suele crear una nueva Branch.

Para crear una nueva rama se ejecuta el comando "git Branch <Nombre>"

```
karen@DESKTOP-9HOHL7M MINGW64 ~/Documents/Estructuras 1/Practica0/Demo/guia (main)
$ git branch CambiarArchivoDeTexto
```

Nota Importante: si al crearla estamos en main, master u otra rama, esta nueva rama se basará en la rama original en la que estábamos. En el caso del ejemplo, se basará en la rama main, es decir, luego de crearla, tendremos exactamente todos los cambios de main.

Una vez creada una rama, debemos navegar hacia ella, esto lo logramos con el comando "git checkout <nombreRama>"

```
karen@DESKTOP-9HOHL7M MINGW64 ~/Documents/Estructuras 1/Practica0/Demo/guia (main)
$ git checkout CambiarArchivoDeTexto
Switched to branch 'CambiarArchivoDeTexto'

karen@DESKTOP-9HOHL7M MINGW64 ~/Documents/Estructuras 1/Practica0/Demo/guia (CambiarArchivoDeTexto)
$
```

En esta rama, podemos llevar a cabo los cambios que destinamos para ella:

Nuevamente, sobre una rama aun si no es la main o master, podemos hacer commits para guardar nuestros cambios en el repositorio local:

```
karen@DESKTOP-9HOHL7M MINGW64 ~/Documents/Estructuras 1/Practica0/Demo/guia (CambiarArchivoDeTexto)
$ git commit -a -m "CambiandoTexto"
[CambiarArchivoDeTexto 2137b1c] CambiandoTexto
 1 file changed, 3 insertions(+), 1 deletion(-)
```

De igual forma, para que nuestra nueva rama sea accedida desde el repositorio remoto, utilizamos "git push":

```
karen@DESKTOP-9HOHL7M MINGW64 ~/Documents/Estructuras 1/Practica0/Demo/guia (CambiarArchivoDeTexto)
$ git push origin CambiarArchivoDeTexto
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 321 bytes | 321.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
remote:
remote: Create a pull request for 'CambiarArchivoDeTexto' on GitHub by visiting:
remote:      https://github.com/kliskis/Guia/pull/new/CambiarArchivoDeTexto
remote:
To https://github.com/kliskis/Guia.git
 * [new branch]      CambiarArchivoDeTexto -> CambiarArchivoDeTexto
```

Merge:

Cuando hemos terminado de trabajar en un cambio, usualmente deberíamos unificar nuestros cambios con el contenido de main o master, para esto, utilizamos la operación **Merge**.

Para hacer un merge, debemos ubicarnos primero en la rama mai/master o cualquier otra sobre la cual queramos añadir los cambios que hay en otra(donde centralizaremos o unificaremos los cambios) con el comando "git checkout",

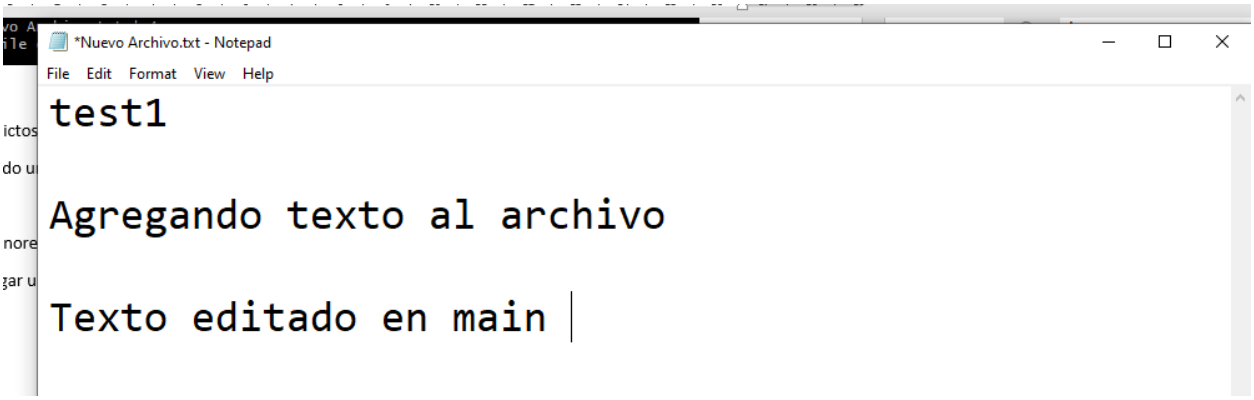
Luego procedemos con el merge, para ello, utilizamos el comando es "git merge <NombreRamaOrigen> -m "Mensaje" "

```
karen@DESKTOP-9HOHL7M MINGW64 ~/Documents/Estructuras 1/Practica0/Demo/guia (main)
$ git merge CambiarArchivoDeTexto
Updating 4681b3a..2137b1c
Fast-forward
 Nuevo Archivo.txt | 4 +++-
 1 file changed, 3 insertions(+), 1 deletion(-)
```

Un merge, necesita hacerse commit y push para que se puedan publicar los cambios sobre el mismo en el Repositorio remoto.

Conflictos:

Cuando un Archivo se modifica en 2 ramas distintas puede existir un conflicto, por ejemplo:

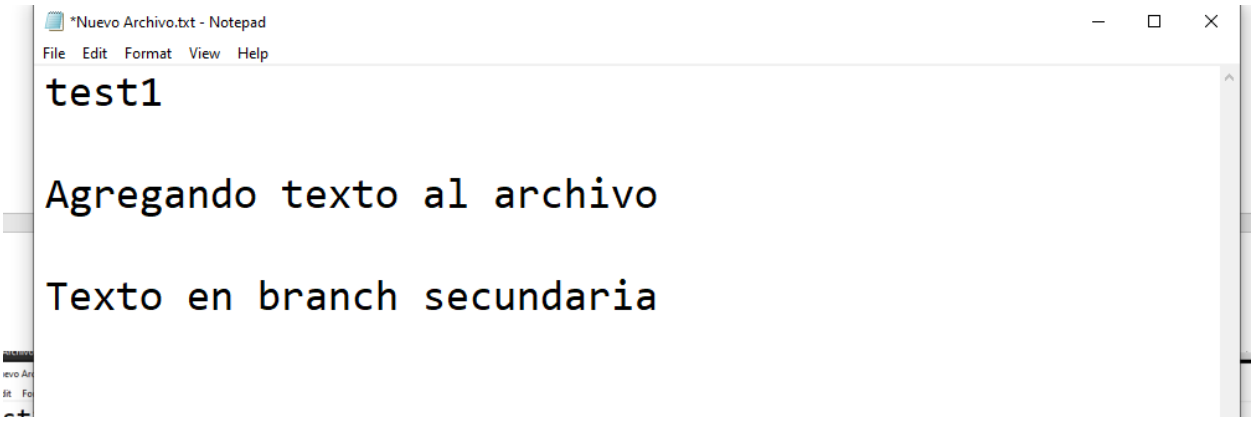



```
karen@DESKTOP-9HOHL7M MINGW64 ~/Documents/Estructuras 1/Practica0/Demo/guia (main)
$ git commit -a -m "agregando texto"
[main 920df8d] agregando texto
 1 file changed, 3 insertions(+), 1 deletion(-)
```

Cambiar a la otra rama:

```
karen@DESKTOP-9HOHL7M MINGW64 ~/Documents/Estructuras 1/Practica0/Demo/guia (main)
$ git checkout CambiarArchivoDeTexto
Switched to branch 'CambiarArchivoDeTexto'
```

Eliminar parte del texto existente y modificar una de las líneas existentes:



Hacer commit sobre esta branch:

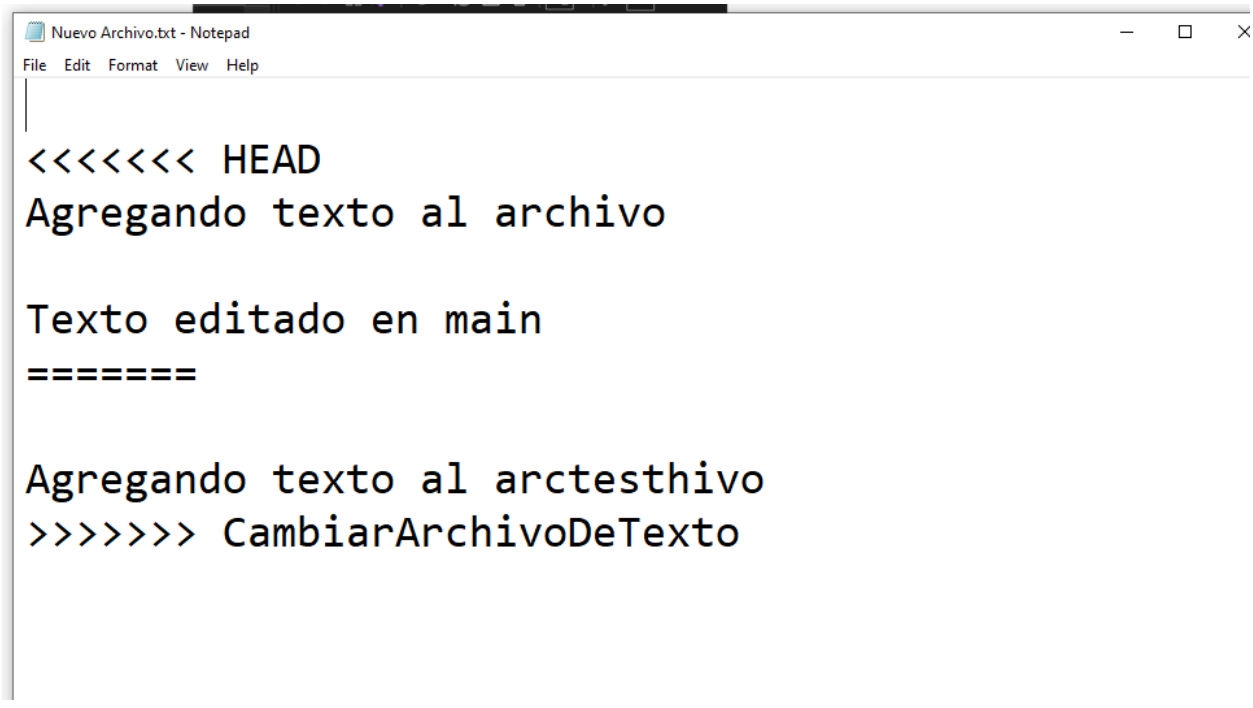
```
karen@DESKTOP-9HOHL7M MINGW64 ~/Documents/Estructuras 1/Practica0/Demo/guia (CambiarArchivoDeTexto)
$ git commit -a -m "cambiar texto 2"
[CambiarArchivoDeTexto 2cce5e9] cambiar texto 2
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Ignored.exe
```

Al intentar hacer el merge, se detectan conflictos:

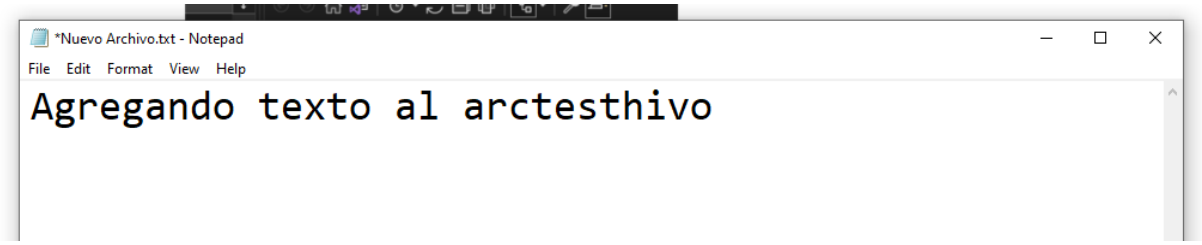
```
karen@DESKTOP-9HOHL7M MINGW64 ~/Documents/Estructuras 1/Practica0/Demo/Guia (mai
n)
$ git merge CambiarArchivoDeTexto -m "Agregando cabios"
```

```
karen@DESKTOP-9HOHL7M MINGW64 ~/Documents/Estructuras 1/Practica0/Demo/Guia (mai
n)
$ git merge CambiarArchivoDeTexto -m "Agregando cambios"
Auto-merging Nuevo Archivo.txt
CONFLICT (content): Merge conflict in Nuevo Archivo.txt
Automatic merge failed; fix conflicts and then commit the result.
```

Al abrir el archivo vemos los conflictos resaltados con un texto agregado por GIT, el cual nos muestra los cambios y nosotros debemos decidir qué cambio debemos dejar editando el archivo:



Seleccionamos la opción que queremos dejar, guardamos el archivo y hacemos commit:



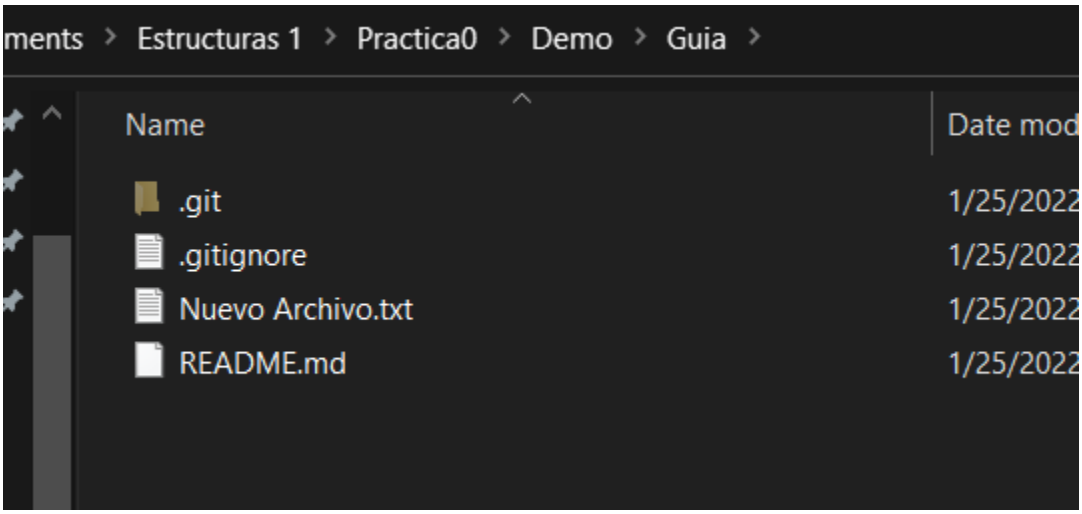

```
karen@DESKTOP-9H0HL7M MINGW64 ~/Documents/Estructuras 1/Practica0/Demo/Guia (mai
n|MERGING)
$ git commit -a -m "escogiendo version final"
[main 5b2fccb] escogiendo version final

karen@DESKTOP-9H0HL7M MINGW64 ~/Documents/Estructuras 1/Practica0/Demo/Guia (mai
n)
$
```

Git ignore:

Las soluciones de Visual Studio, son muy pesadas ya que almacenan archivos de compilación y otros archivos locales que se utilizan para la ejecución dependiendo de la instalación de cada computadora. No nos interesa subir estos archivos al repositorio por lo que utilizamos **gitignore**.

Agregar un archivo llamado **.gitignore** en la raíz del proyecto como se muestra a continuación:



Estando en la rama master agregar el git ignore usando este comando

"git add .gitignore"

```
karen@DESKTOP-9HOHL7M MINGW64 ~/Documents/Estructuras 1/Practica0/Demo/guia (main)
$ git add .gitignore
```

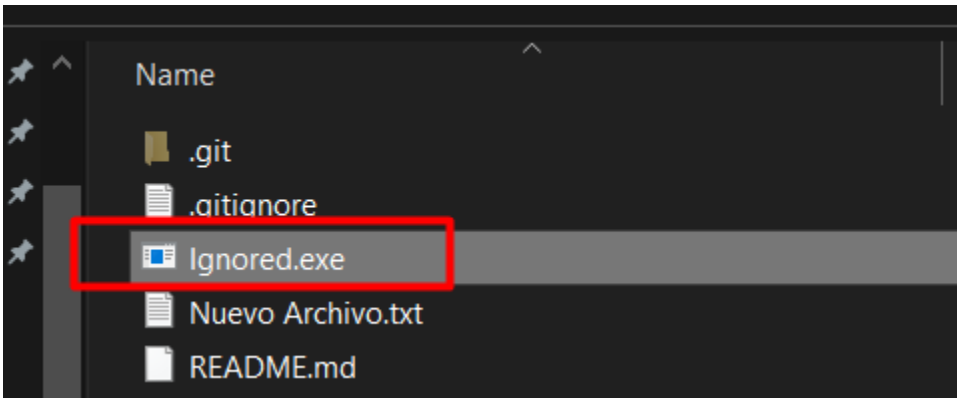
Abrir el archivo y agregar las extensiones que se desea ignorar:

```
1  *.com
2  *.class
3  *.dll
4  *.exe
5  *.pdb
6  *.dll.config
7  *.cache
8  *.suo
```

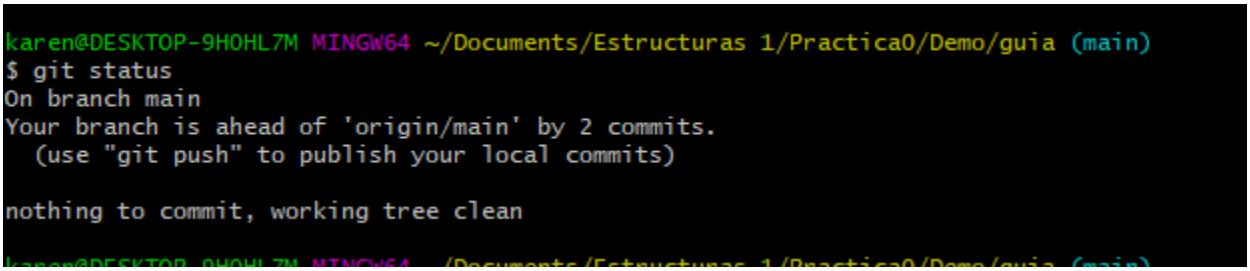
Guardar cambios y hacer commit:

```
karen@DESKTOP-9HOHL7M MINGW64 ~/Documents/Estructuras 1/Practica0/Demo/guia (main)
$ git commit -a -m "Agregando extensiones a ignorar"
[main 73dc697] Agregando extensiones a ignorar
 1 file changed, 8 insertions(+)
 create mode 100644 .gitignore
```

Agregar un archivo de esa extensión, y hacer un git status:

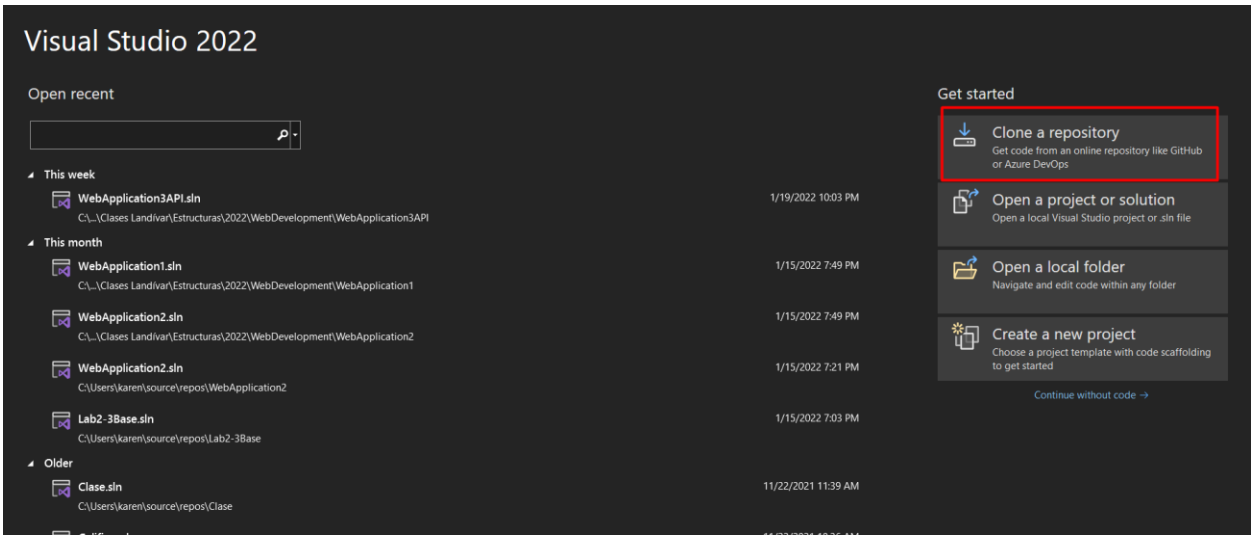


No se detectaron archivos nuevos a agregar como en los casos anteriores:

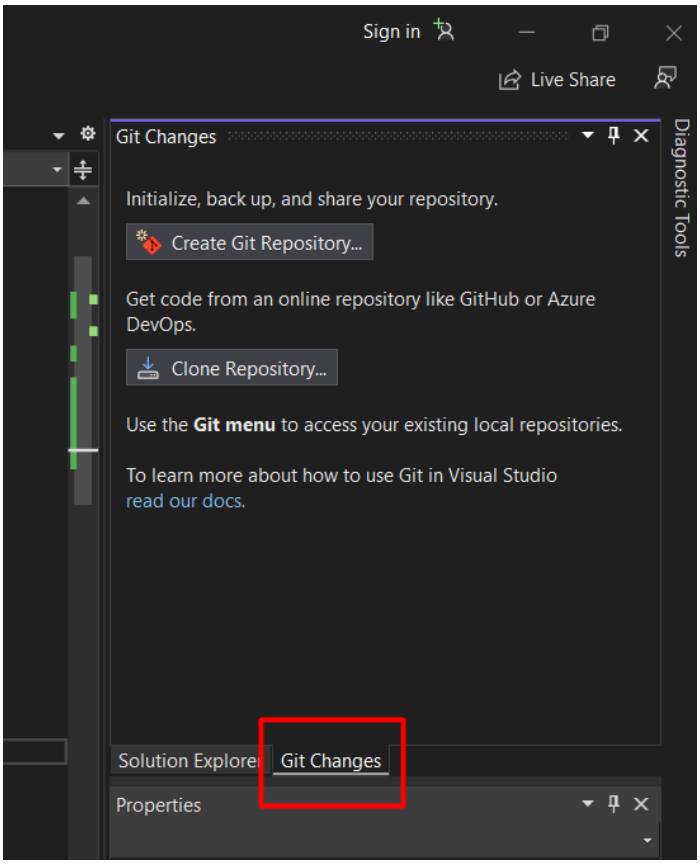


Desde Visual Studio:

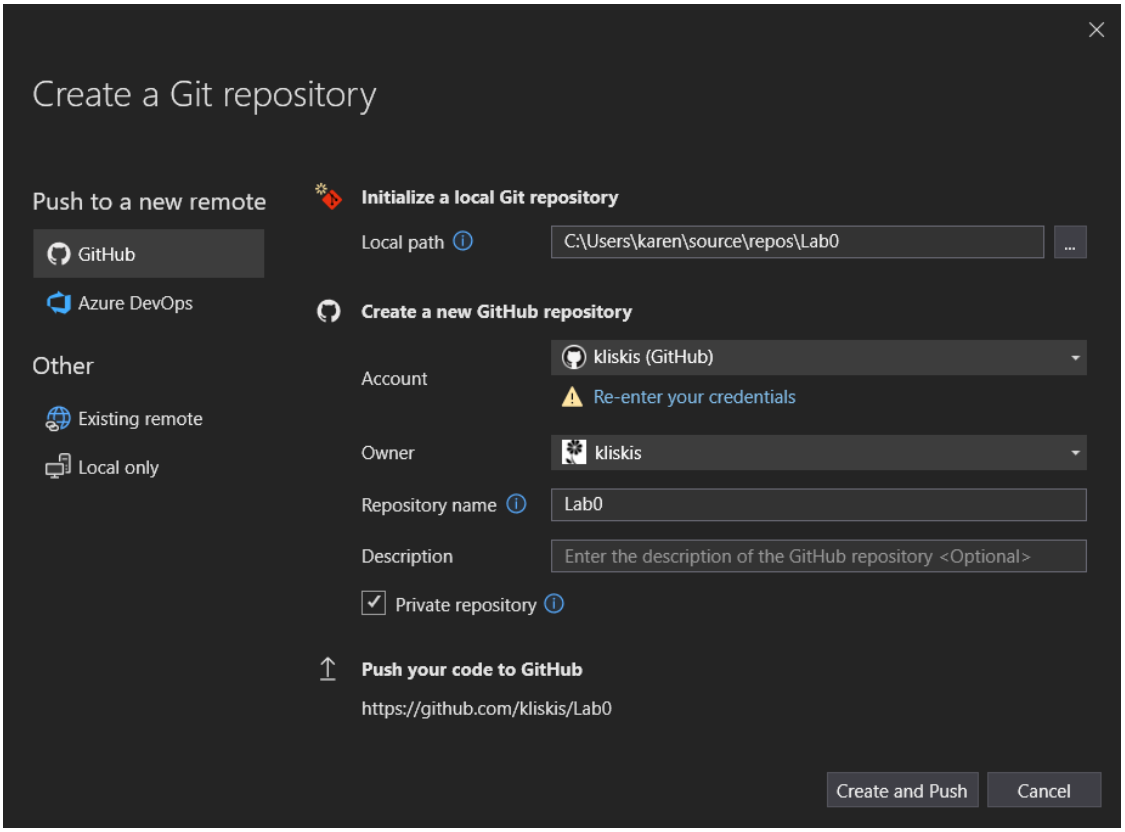
Desde VS también se puede clonar un repositorio:



Desde Visual Studio:

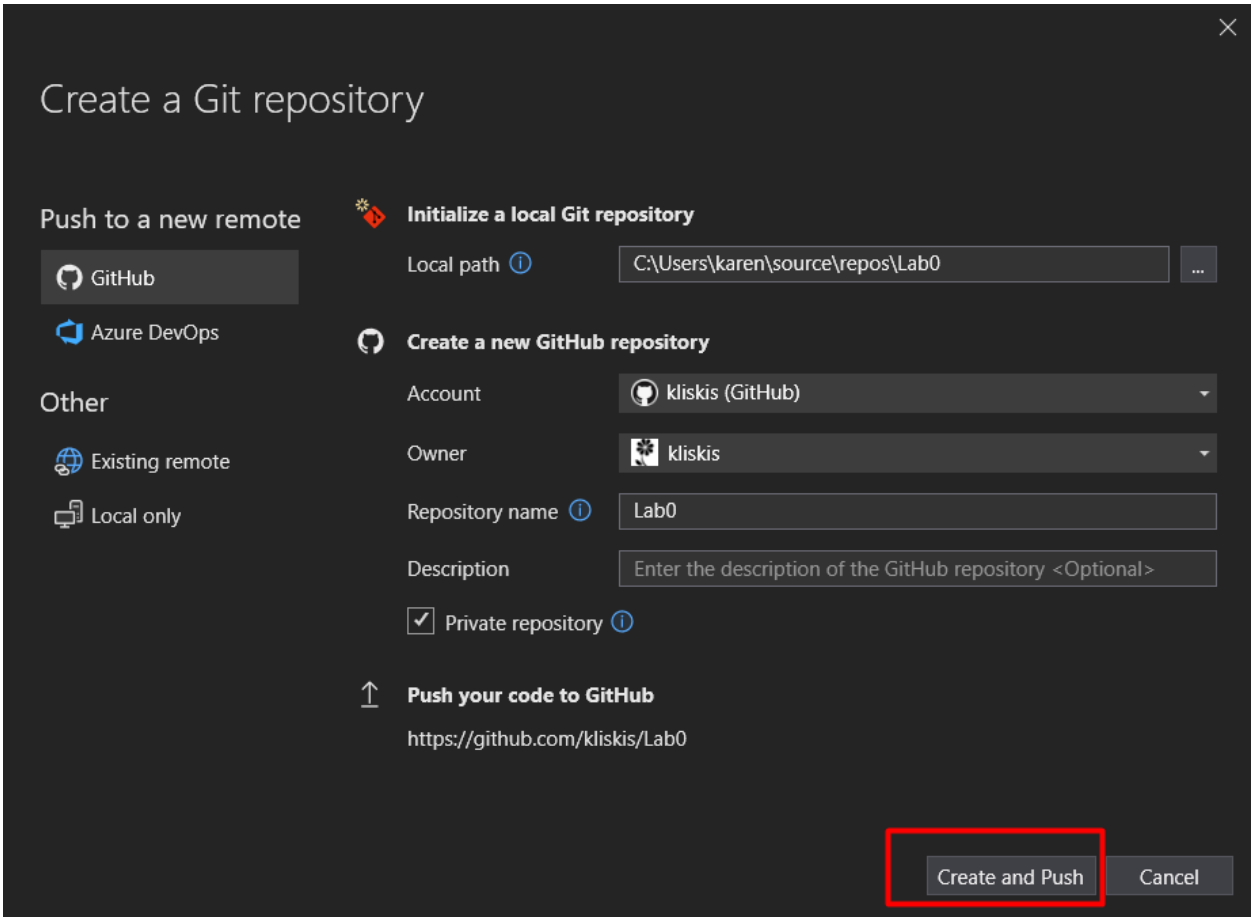


Creando un repositorio:

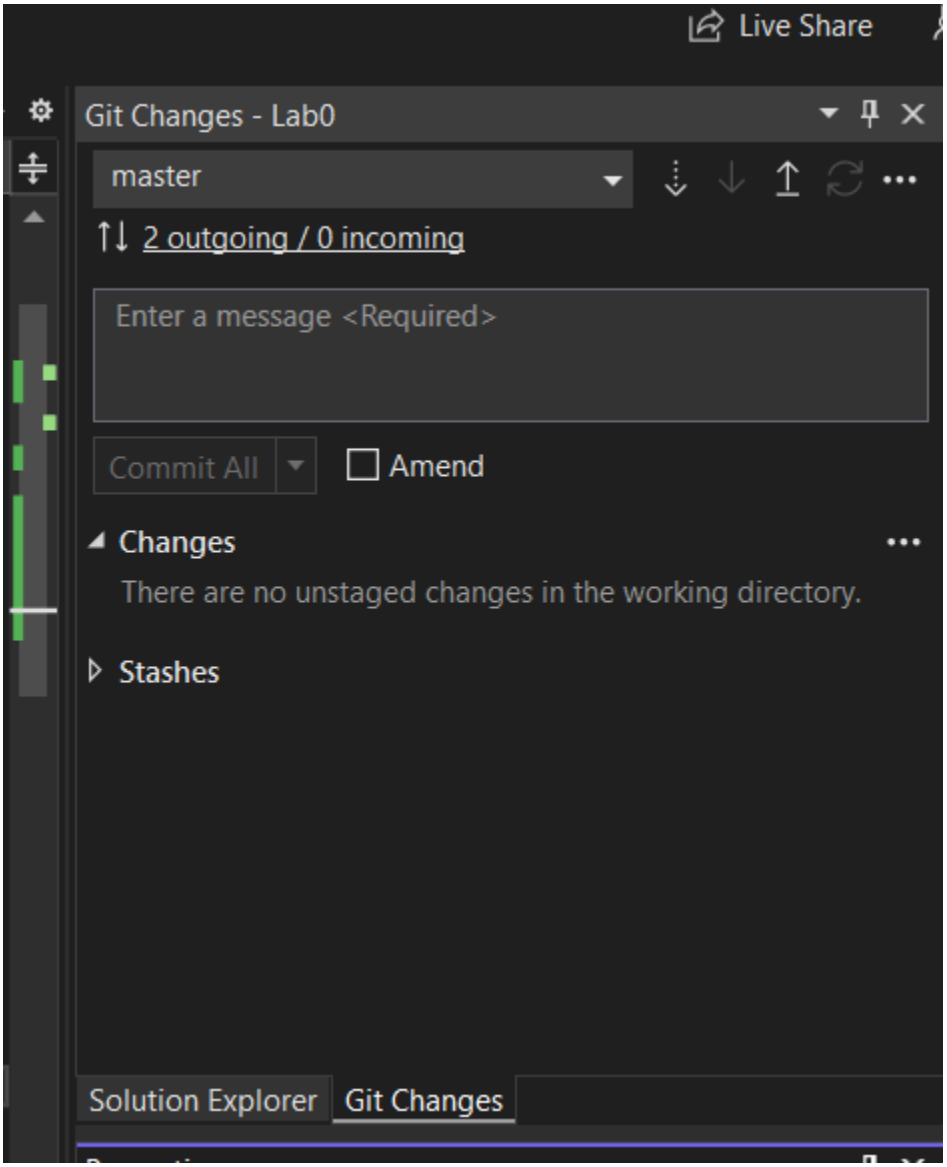

Create a Git repository
Push to a new remote
GitHub
Azure DevOps
Other
Existing remote
Local only
Initialize a local Git repository
Local path
C:\Users\karen\source\repos\Lab0
...
Create a new GitHub repository
Account
kliskis (GitHub)
Re-enter your credentials
Owner
kliskis
Repository name
Lab0
Description
Enter the description of the GitHub repository <Optional>
Private repository
Push your code to GitHub
https://github.com/kliskis/Lab0
Create and Push
Cancel

Credenciales:

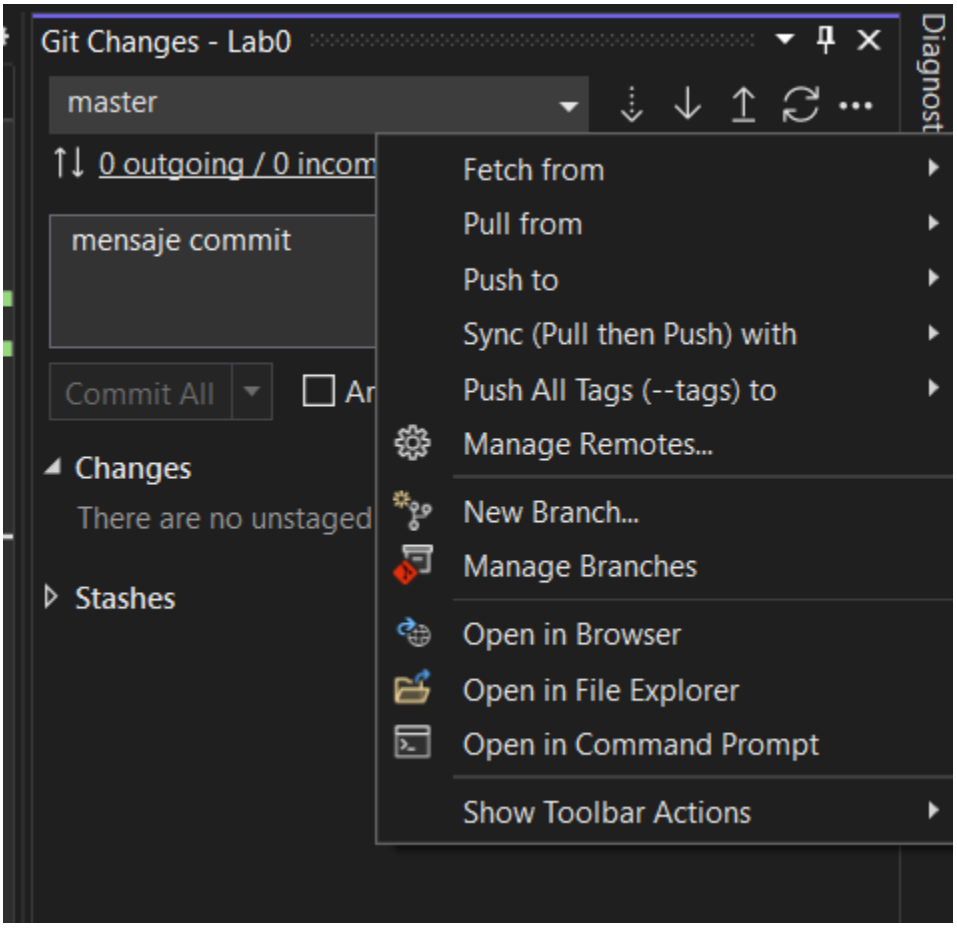
Create and push


Create a Git repository
Push to a new remote
GitHub
Azure DevOps
Other
Existing remote
Local only
Initialize a local Git repository
Local path
C:\Users\karen\source\repos\Lab0
...
Create a new GitHub repository
Account
kliskis (GitHub)
Owner
kliskis
Repository name
Lab0
Description
Enter the description of the GitHub repository <Optional>
Private repository
Push your code to GitHub
https://github.com/kliskis/Lab0
Create and Push
Cancel

Commits

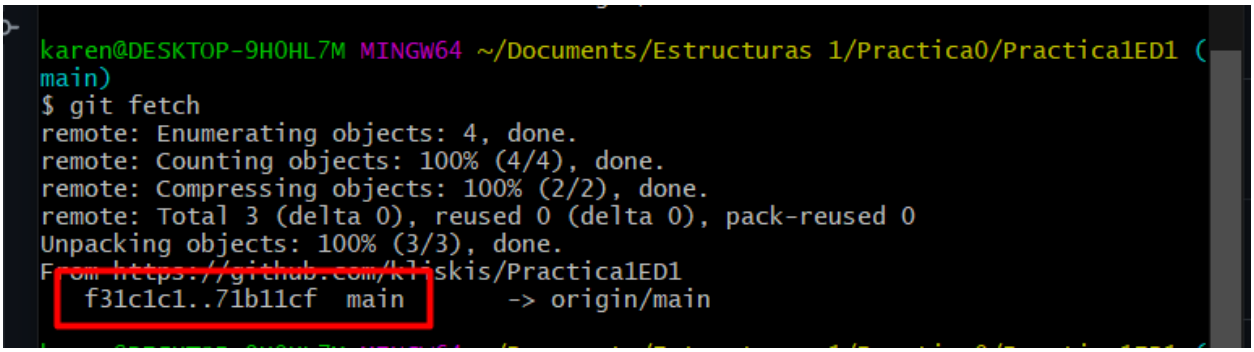


Todas las acciones del bash


Git Changes - Lab0
master
0 outgoing / 0 incom
mensaje commit
Commit All
Changes
There are no unstaged
Stashes
Fetch from
Pull from
Push to
Sync (Pull then Push) with
Push All Tags (--tags) to
Manage Remotes...
New Branch...
Manage Branches
Open in Browser
Open in File Explorer
Open in Command Prompt
Show Toolbar Actions

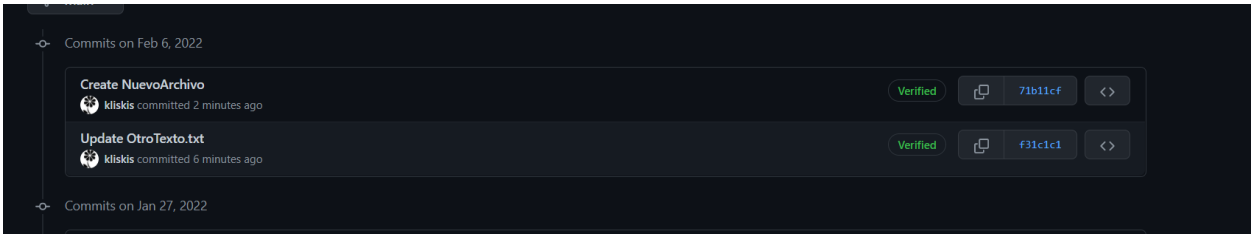
Obtener Cambios Remotos:

Si hemos realizado cambios desde Github Web o, si otro miembro del equipo ha realizado cambios y les ha hecho push, primero debemos verificar si existen nuevos cambios en el repositorio, para esto utilizamos el comando "Git Fetch":

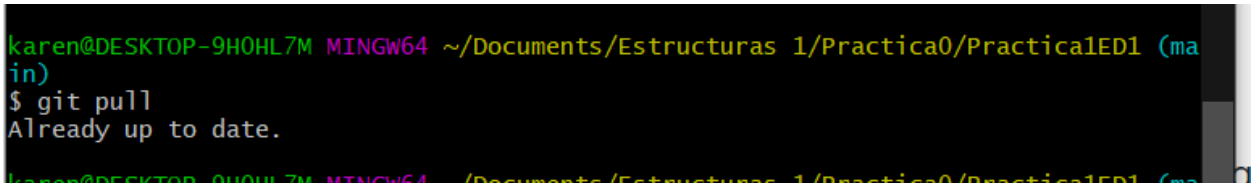


Es muy importante que podemos ver el ID de los últimos commits y verificar si tenemos el último o no.

Por ejemplo, estos commits fueron realizados solamente en el repositorio remoto



Por lo que para obtener la última versión debemos ejecutar el comando "git Pull":



También es posible solamente hacer git pull sin fetch, en caso no queramos verificar los cambios presentes. Hace merge automáticamente de los cambios del remote repo al local repo.

Importante:

- Git pull hace 2 operaciones, la primera es verificar los cambios del repositorio remoto, es decir, ejecuta el equivalente a un git fetch, luego toma los cambios y hace un merge a los cambios locales.
- Dado que ejecuta un merge, podrían existir conflictos, en este caso se resuelven exactamente igual a como se ha abordado en la sección de conflictos.
- Finalmente, ejecuta un commit local automático indicando que se ha hecho merge con ciertos cambios del repositorio remoto.

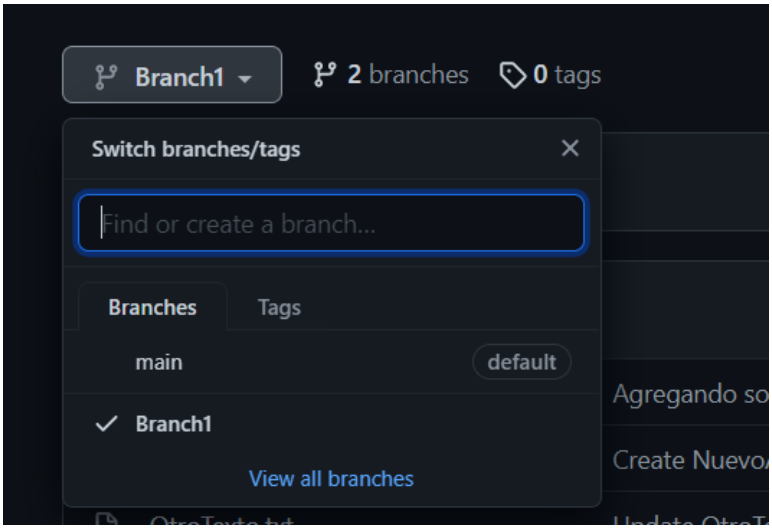
Considerar antes de Hacer pull:

Es recomendable ejecutar un "git status" antes de hacer pull de cambios remotos, ya que para hacer un pull correcto, todos nuestros cambios locales deberán estar dentro de un commit, si no lo hemos hecho, nos forzará o a **deshacerlos** o a hacer un commit.

Obtener Ramas Remotas:

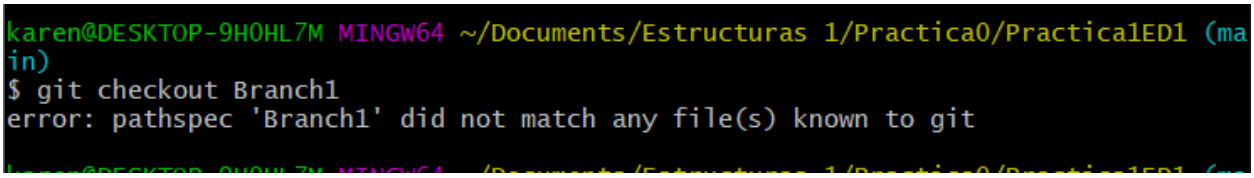
Si hemos creado ramas remotas, no podremos cambiar a ellas haciendo un git checkout, si no hacemos primero un "git Fetch"

Por ejemplo se ha creado esta nueva Branch desde otro equipo:



Por el momento podemos verla solamente desde el sitio de github ya que está disponible de manera remota únicamente.

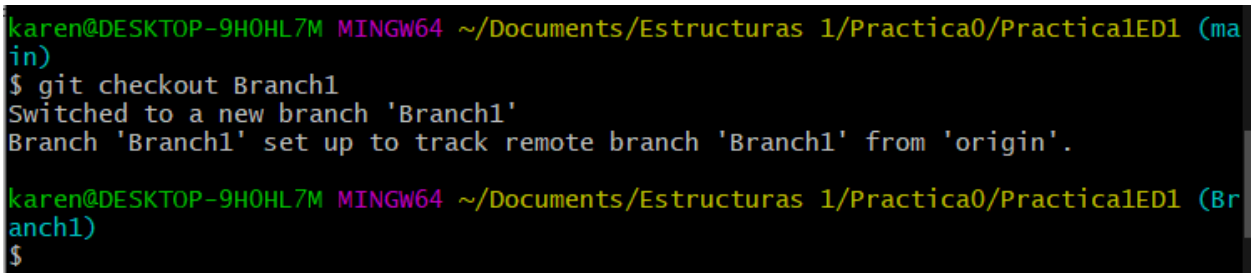
Si intentamos hacer "git checkout" hacia esa rama que no hemos creado en nuestro equipo, nos dará un error:



Si ejecutamos el comando "git fetch" podremos verla y luego hacer checkout a ella:

Al ejecutar git fetch, se puede observar que encuentra la rama que hemos creado.

Ahora sí, podemos ejecutar el git checkout para cambiarnos a esa rama nueva:



Sobrescribir cambios locales:

Pueden existir casos donde no queramos conservar nuestros cambios locales, y caerle encima con los cambios que están en el repositorio remoto.

Para esto debemos tomar en cuenta que **no debemos hacer commit** de nuestros cambios locales antes de hacer pull.

Y para no tener problema a la hora de ejecutar el comando "git pull" debemos agregar un parámetro a este comando:

Git pull –force o git pull -f:

```
karen@DESKTOP-9H0HL7M MINGW64 ~/Documents/Estructuras 1/Practica0/Practica1ED1 (Br
anch1)
$ git pull -f
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/kliskis/Practica1ED1
   71b11cf..47a4fcf  Branch1    -> origin/Branch1
Updating 71b11cf..47a4fcf
Fast-forward
 OtroTexto.txt | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Deshacer un commit:

También es posible que nos encontremos en la situación, que ya hemos hecho algún commit local y queramos deshacerlo antes de hacer push.

En este caso hemos agregado un commit que vamos a deshacer:

```
karen@DESKTOP-9H0HL7M MINGW64 ~/Documents/Estructuras 1/Practica0/Practica1ED1 (Br
anch1)
$ git commit -a -m "Commit 1"
[Branch1 5d0ea7f] Commit 1
 1 file changed, 1 insertion(+), 1 deletion(-)

karen@DESKTOP-9H0HL7M MINGW64 ~/Documents/Estructuras 1/Practica0/Practica1ED1 (Br
anch1)
$ git commit -a -m "Commit para deshacer"
[Branch1 c93dec2] Commit para deshacer
 1 file changed, 2 insertions(+), 1 deletion(-)
```

Para ello, utilizaremos el comando: git reset --hard HEAD~1

```
karen@DESKTOP-9H0HL7M MINGW64 ~/Documents/Estructuras 1/Practica0/Practica1ED1 (Br
anch1)
$ git reset --hard HEAD~1
HEAD is now at 5d0ea7f Commit 1
```

Hemos regresado al commit 1, y si verificamos el archivo, el cambio no está:

Antes:

```
1
2
3 tex<textointermedio>to6 test2asdfasdf
4
5 TEst A
6 test - Agregado commit 1
```

```
1
2
3 tex<textointermedio>to6 test2asdfasdf
4
5 TEst A
6 test - Agregado commit 1
7 Cambio para deshacer
```

Después del reset:

```
1
2
3 tex<textointermedio>to6 test2asdfasdf
4
5 TEst A
6 test - Agregado commit 1
```