

# Practica de GIT y GITHUB

Introducción a la Geoinformática

17/05/2021

## COMANDOS BÁSICOS:

Al abrir Git Bash te abre por default la carpeta donde están los archivos de tu usuario

```
MINGW64:/c/Users/Paulina
Paulina@DESKTOP-9A4451G MINGW64 ~
$ |
```

**pwd:** Nos muestra la ruta de carpetas en la que te encuentras ahora mismo.

```
MINGW64:/c/Users/Paulina
Paulina@DESKTOP-9A4451G MINGW64 ~
$ pwd
/c/Users/Paulina
```

Si se quiere regresar una carpeta se aplica el cambio de directorio y los dos puntos

```
Paulina@DESKTOP-9A4451G MINGW64 ~
$ cd ..

Paulina@DESKTOP-9A4451G MINGW64 /c/Users
$ pwd
/c/Users
```

Para regresar la carpeta por default

```
Paulina@DESKTOP-9A4451G MINGW64 /c/Users
$ cd

Paulina@DESKTOP-9A4451G MINGW64 ~
$ pwd
/c/Users/Paulina
```

Queremos posicionarnos en la carpeta Documentos, necesitamos conocer el nombre exacto de esta carpeta. Con el comando **ls** podemos conocer el listado de carpetas y archivos.

```
Paulina@DESKTOP-9A4451G MINGW64 ~
$ ls
1.0
'3D Objects'/
3_Matplotlib.ipynb
AppData/
Basemap.ipynb
CmapToolsLogs/
'Configuración local'@
Contacts/
Cookies@
'Creative Cloud Files'/
'Datos de programa'@
Desktop/
Documents/
Downloads/
Dropbox/
Ejercicio_1.ipynb
```

Una vez identificado como se debe escribir cambiamos la dirección utilizando el comando **cd** *nombre de la carpeta*.

```
Paulina@DESKTOP-9A4451G MINGW64 ~
$ cd Documents

Paulina@DESKTOP-9A4451G MINGW64 ~/Documents
$ pwd
/c/Users/Paulina/Documents
```

También se puede usar la dirección completa utilizando el comando **cd** y entre comillas la dirección donde esta la carpeta, incluso si se encuentra en otro disco o USB.

```
MINGW64:/g/GEOESPACIAL A.C/Geoinformatica_Asignatura/Git y GitHub
Paulina@DESKTOP-9A4451G MINGW64 ~
$ cd "G:\GEOESPACIAL A.C\Geoinformatica_Asignatura\Git y GitHub"

Paulina@DESKTOP-9A4451G MINGW64 /g/GEOESPACIAL A.C/Geoinformatica_Asignatura/Git
y GitHub
```

Agregar una carpeta en Documentos llamada practica\_git con el comando **mkdir**

```
Paulina@DESKTOP-9A4451G MINGW64 ~/Documents
$ mkdir practica_git
```

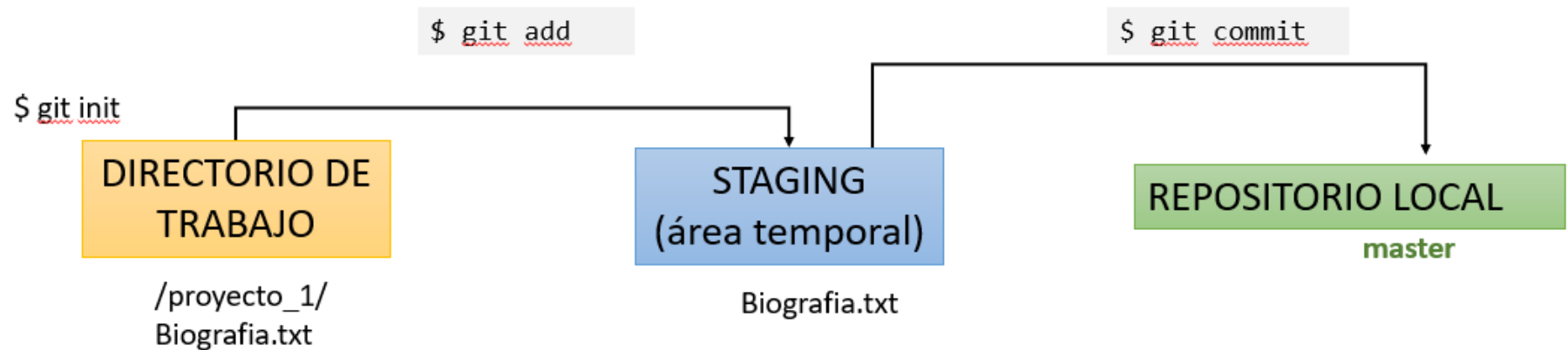
Otros comandos:

- **touch:** Nos permite crear archivos (por ejemplo, `touch archivo.txt`).
- **rm:** Nos permite borrar un archivo o carpeta (por ejemplo, `rm archivo.txt`). Mucho cuidado con este comando, puedes borrar todo tu disco duro si no sabes en qué dirección estas.
- **cat:** Ver el contenido de un archivo (por ejemplo, `cat nombre-archivo.txt`).
- **ls:** Nos permite cambiar ver los archivos de la carpeta donde estamos ahora mismo. Podemos usar uno o más argumentos para ver más información sobre estos archivos (los argumentos pueden ser `--` + el nombre del argumento o `-` + una sola letra o shortcut por cada argumento).
  - `ls -a`: Mostrar todos los archivos, incluso los ocultos.
  - `ls -l`: Ver todos los archivos como una lista.
- **cd:** Nos permite navegar entre carpetas.
  - `cd /`: Ir a la ruta principal:
  - `cd carpeta/subcarpeta`: Navegar a una ruta dentro de la carpeta donde estamos ahora mismo.
  - `cd ..` (`cd` + dos puntos): Regresar una carpeta hacia atrás.
  - Si quieres referirte al directorio en el que te encuentras ahora mismo puedes usar `cd .` (`cd` + un punto).
- **history:** Ver los últimos comandos que ejecutamos y un número especial con el que podemos repetir su ejecución.
- **clear:** Para limpiar la terminal. También podemos usar los atajos de teclado `Ctrl + L` O `Command + L`.



# Ciclo básico de trabajo en Git

Para iniciar un repositorio, o sea, activar el sistema de control de versiones de Git, solo debes ejecutar el comando `$ git init`

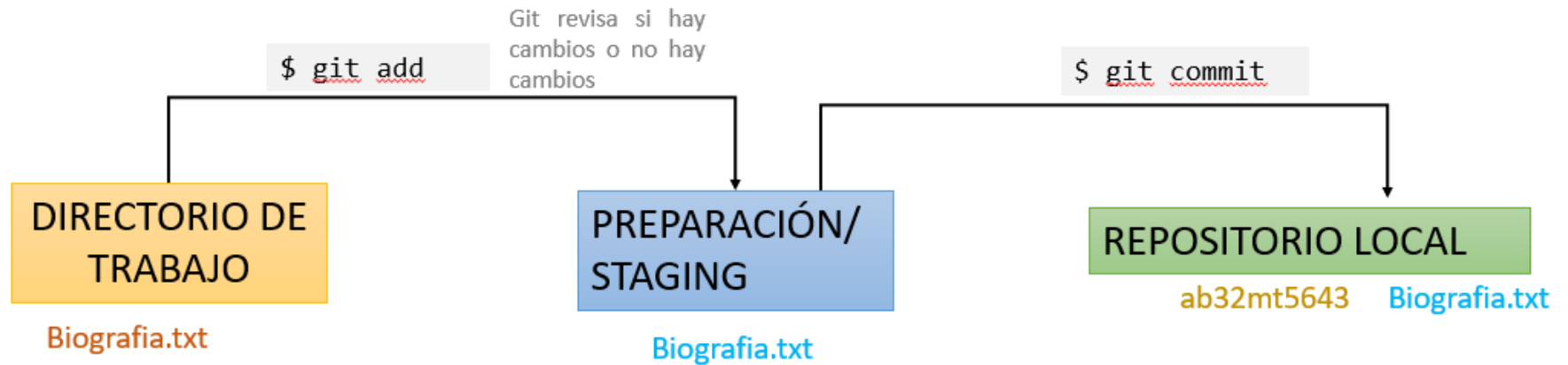


`$ git init`: Crea un área en memoria RAM (Staging), área en donde al principio se agregan los cambios. También se crea un Repositorio (carpeta `.Git`), ahí van a estar todos los cambios al final del proyecto.

`$ git add`: Mueve archivos al estado Staged. Podemos usar `git nombre-del-archivo-o-carpeta` para añadir archivos y carpetas individuales o `git add -A` para mover todos los archivos de nuestro proyecto .

`$ git commit`: Mueve archivos a master. Git nos pedirá que dejemos un mensaje para recordar los cambios que hicimos y podemos usar el argumento `-m` para escribirlo (`git commit -m "mensaje"`).

Cuando trabajamos en Git, nuestros archivos pueden “vivir” entre diferentes estados



**Untracked:** Archivos que NO viven dentro de Git, solo en el disco duro. Sin rastrear porque todavía no le hemos dado `git add`

**Tracked:** Archivos que viven dentro de Git. Han sido rastreados y son parte de **Staging**. Pasan a este estado gracias al `$git add` y `$git commit`.

## Configurando nuestro entorno en Git

Configurar Git, con nuestro nombre de usuario y correo para darnos una identidad y ayuda a saber quién hizo los cambios cuando trabajas con más usuarios.

```
Paulina@DESKTOP-9A4451G MINGW64 ~/Documents/practica_git (master)
$ git config --global user.name "PaulinaPC"
```

```
Paulina@DESKTOP-9A4451G MINGW64 ~/Documents/practica_git (master)
$ git config --global user.email "miemail@gmail.com"
```

Para darnos cuenta si se hicieron los cambios:

```
Paulina@DESKTOP-9A4451G MINGW64 ~/Documents/practica_git (master)
$ git config --list
```

En la carpeta llamada practica\_git copia todos los documentos pertenecientes al drive sobre esta practica

Para crear un repositorio se tienen que ubicar en la carpeta principal donde están los archivos, en este caso en la carpeta practica\_git. Para crear un repositorio se utiliza el comando **git init**.

Para ver si se creo este repositorio en la carpeta con el comando **ls -la** podemos ver archivos ocultos pertenecientes al repositorio creado por git.

```
Paulina@DESKTOP-9A4451G MINGW64 ~/Documents/practica_git
$ git init
Initialized empty Git repository in C:/Users/Paulina/Documents/practica_git/.git/

Paulina@DESKTOP-9A4451G MINGW64 ~/Documents/practica_git (master)
$ ls -la
total 96
drwxr-xr-x 1 Paulina 197121 0 May 15 22:47 ./
drwxr-xr-x 1 Paulina 197121 0 May 15 22:46 ../
drwxr-xr-x 1 Paulina 197121 0 May 15 22:47 .git/
```

El comando **git status** permite conocer si hay archivos en modo untracked, o en staging o si ya se les hizo un commit. En este caso, nos dice que tenemos un archivo llamado novela.docx que no se ha agregado o hecho commit.

```
Paulina@DESKTOP-9A4451G MINGW64 ~/Documents/practica_git (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        novela.docx
        ~$novela.docx

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

Para agregar el archivo a staging utilizamos **git add** y volvemos a checar el estatus que nos dirá que existe un archivo .docx al que se le puede hacer commit y enviarlo al repositorio local.

```
Paulina@DESKTOP-9A4451G MINGW64 ~/Documents/practica_git (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   novela.docx

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        ~$novela.docx
```

- Por ejemplo, si llegamos a equivocarnos y ese archivo no era el que se tenía que subir, se puede sacar con **git rm --cached**

```
Paulina@DESKTOP-9A4451G MINGW64 ~/Documents/practica_git (master)
$ git rm --cached novela.docx
rm 'novela.docx'
```

Si revisas tu carpeta de practica\_git notarás que el archivo novela.docx sigue existiendo ya que solo se eliminó de git y no de nuestra carpeta en memoria C. Incluso lo puedes comprobar con **git status**

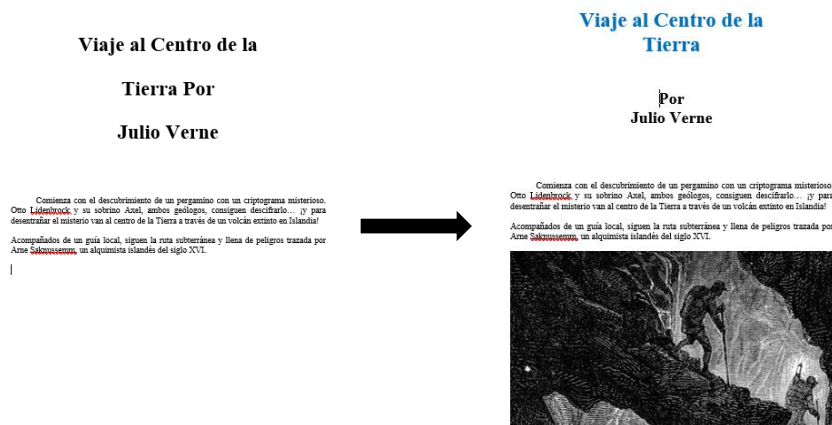
Para volver a subir el archivo a git, vuélvelo agregar con **git add**.

## Flujo de trabajo en Git

Recuerda que git add agrega el archivo a nuestra zona de preparación (staging) pero aun no se ha subido al repositorio local, para hacerlo utiliza **git commit** junto con un mensaje que incluya a que se refiere ese commit.

```
Paulina@DESKTOP-9A4451G MINGW64 ~/Documents/practica_git (master)
$ git commit -m "Este es el primer commit de esta practica"
[master (root-commit) 39ed1bc] Este es el primer commit de esta practica
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 novela.docx
```

Ya tenemos en el repositorio nuestro archivo novela.docx. Pero ahora resulta que modificamos este archivo al agregarle una imagen a la portada y mejorando la edición:



Si volvemos a correr el comando **git status** vemos que Git reconoció estas modificaciones y sugiere agregar las modificaciones.

```
Paulina@DESKTOP-9A4451G MINGW64 ~/Documents/practica_git (master)
$ git status
On branch master
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:   novela.docx
```

Agregamos nuevamente el archivo novela.docx. Nota primero se agrega al staging y después se sube al repositorio.

```
Paulina@DESKTOP-9A4451G MINGW64 ~/Documents/practica_git (master)
$ git add novela.docx

Paulina@DESKTOP-9A4451G MINGW64 ~/Documents/practica_git (master)
$ git commit -m "Cambios a la portada"
[master ccd135b] Cambios a la portada
1 file changed, 0 insertions(+), 0 deletions(-)
rewrite novela.docx (74%)
```

Para conocer la historia de nuestro archivo de lo que se ha subido a Git se utiliza **git log**.

```
Paulina@DESKTOP-9A4451G MINGW64 ~/Documents/practica_git (master)
$ git log novela.docx
commit ccd135b833a92b94cc4c02d5bb30f2bdb6f6e6e1 (HEAD -> master)
Author: PaulinaPC <paulina.geopc@gmail.com>
Date: Sat May 15 23:53:25 2021 -0500

    Cambios a la portada

commit 39ed1bc879d1b643718c3e450517a67a60e736ff
Author: PaulinaPC <paulina.geopc@gmail.com>
Date: Sat May 15 23:26:17 2021 -0500

    Este es el primer commit de esta practica
```

Para conocer exactamente qué cambios se hicieron y el usuario que los hizo se utiliza **git show**

```
Paulina@DESKTOP-9A4451G MINGW64 ~/Documents/practica_git (master)
$ git show novela.docx
commit ccd135b833a92b94cc4c02d5bb30f2bdb6f6e6e1 (HEAD -> master)
Author: PaulinaPC <paulina.geopc@gmail.com>
Date: Sat May 15 23:53:25 2021 -0500

    Cambios a la portada

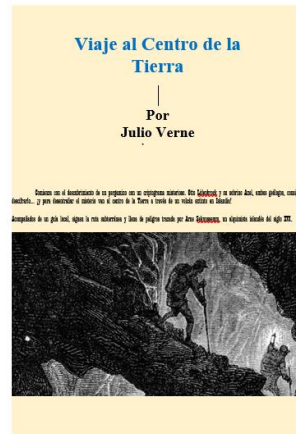
diff --git a/novela.docx b/novela.docx
index 4e1443d..b183bcd 100644
--- a/novela.docx
+++ b/novela.docx
@@ -1,4 +1,6 @@
-Viaje al Centro de la Tierra Por
+Viaje al Centro de la Tierra
+
+Por
Julio Verne

@@ -6,3 +8,4 @@ Julio Verne
    Comienza con el descubrimiento de un pergamino con un criptograma misterioso. Otto Lidenbrock y su sobrino Axel, ambos geólogos, consiguen descifrarlo... ¡y para desentrañar el misterio van al centro de la Tierra a través de un volcán extinto en Islandia!
    Acompañados de un guía local, siguen la ruta subterránea y llena de peligros trazada por Arne Saknussemm, un alquimista islandés del siglo XVI.
```



## - ¿Cómo regresar a una versión anterior?

Se volvió a modificar la portada, para ver si podría quedar de mejor forma; en este caso quedo de la siguiente manera:



Estos nuevos cambios se agregan y se suben al repositorio local del Git. Volvemos aplicar **git log** para ver el historial que ha tenido el archivo novela.docx. Notamos que hemos subido tres tipos de versiones. Pero la versión final se encuentra en el commit tres, esto lo podemos saber porque después del número de identificación del commit aparece la instrucción (HEAD -> master)

```
Paulina@DESKTOP-9A4451G MINGW64 ~/Documents/practica_git (master)
$ git add novela.docx

Paulina@DESKTOP-9A4451G MINGW64 ~/Documents/practica_git (master)
$ git commit -m "Commit tres sobre cambios de color y letra a la portada"
[master 57856e5] Commit tres sobre cambios de color y letra a la portada
1 file changed, 0 insertions(+), 0 deletions(-)

Paulina@DESKTOP-9A4451G MINGW64 ~/Documents/practica_git (master)
$ git log novela.docx
commit 57856e52f1dca94240c95081fdd540a07105cd4c (HEAD -> master)
Author: PaulinaPC <paulina.geopc@gmail.com>
Date: Sun May 16 00:30:16 2021 -0500

    Commit tres sobre cambios de color y letra a la portada

commit ccd135b833a92b94cc4c02d5bb30f2bdb6f6e6e1
Author: PaulinaPC <paulina.geopc@gmail.com>
Date: Sat May 15 23:53:25 2021 -0500

    Cambios a la portada

commit 39ed1bc879d1b643718c3e450517a67a60e736ff
Author: PaulinaPC <paulina.geopc@gmail.com>
Date: Sat May 15 23:26:17 2021 -0500

    Este es el primer commit de esta practica
```

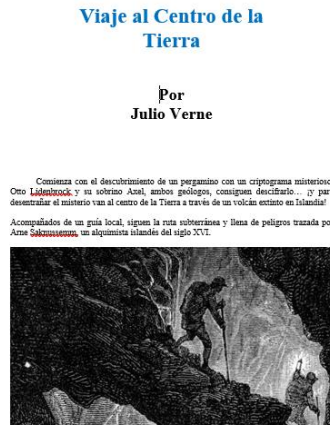
De estas tres versiones, la portada que mejor convenció fue del segundo commit cuando se agregó la imagen y se modifico el estilo del título.

Para regresar a la segunda versión se utiliza **git reset**. Este comando maneja dos tipos de reset, el duro y el suave (hard/soft). Si se utiliza el **git reset "id del commit" --hard** TODO vuelve al estado anterior, mientras que el soft deja en staging la versión y no la elimina por completo, este comando se utiliza si en algún momento decides utilizar esa versión.

En esta práctica utilizaremos el Hard ya que no nos interesa trabajar con la portada amarilla, para esto copiamos con click derecho el id del commit de la versión dos y lo pegamos entre **git reset** y **--hard**:

```
Paulina@DESKTOP-9A4451G MINGW64 ~/Documents/practica_git (master)
$ git reset ccd135b833a92b94cc4c02d5bb30f2bdb6f6e6e1 --hard
HEAD is now at ccd135b Cambios a la portada
```

Para conocer si se aplicaron los cambios, abrimos el archivo Word de novela.docx, y notaremos que se ha cambiado a la portada anterior.



## - Creando ramas o branches de Git

Las ramas son la forma de hacer cambios en nuestro proyecto sin afectar el flujo de trabajo de la rama principal. Se utilizan cuando se quiere trabajar en una sección muy específica o simplemente experimentar.

La rama principal se llama master.

```
Paulina@DESKTOP-9A4451G MINGW64 ~/Documents/practica_git (master)
```

Cuando se crea una rama lo que hace es crear una copia del último commit, y todos los cambios que se hagan en esta rama no se van a ver reflejados en la rama principal (master) hasta que no se fusione con un proceso llamado **merge**.

Para crear una rama, se utiliza el comando **git branch** más el nombre que elijamos para esa rama.

```
Paulina@DESKTOP-9A4451G MINGW64 ~/Documents/practica_git (master)
$ git branch contenido
```

Aunque ya se creo una rama, podemos observar que seguimos trabajando en la principal que es master, para movernos y trabajar en la rama llamada contenido se utiliza **git checkout**.

```
Paulina@DESKTOP-9A4451G MINGW64 ~/Documents/practica_git (master)
$ git checkout contenido
Switched to branch 'contenido'

Paulina@DESKTOP-9A4451G MINGW64 ~/Documents/practica_git (contenido)
$ |
```

Una vez posicionados en la rama contenidos trabajaremos con una copia de la versión de novela.docx pero sin afectar al archivo original que se encuentra en la rama master.

Abrimos el archivo Word de novela.docx y anexamos la información de contenido (en este caso capítulos). Una vez hecho esto guardamos los cambios y los subimos a la rama contenido.

```
Paulina@DESKTOP-9A4451G MINGW64 ~/Documents/practica_git (contenido)
$ git status
On branch contenido
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:   novela.docx

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Viaje al centro de la tierra-original.docx
        critica.docx
        imagen.jpg
        ~$aje al centro de la tierra-original.docx

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

Paulina@DESKTOP-9A4451G MINGW64 ~/Documents/practica_git (contenido)
$ git commit -am "Este es el archivo con capitulo 1 y 2 de Juanito"
[contenido d976c2e] Este es el archivo con capitulo 1 y 2 de Juanito
1 file changed, 0 insertions(+), 0 deletions(-)
```

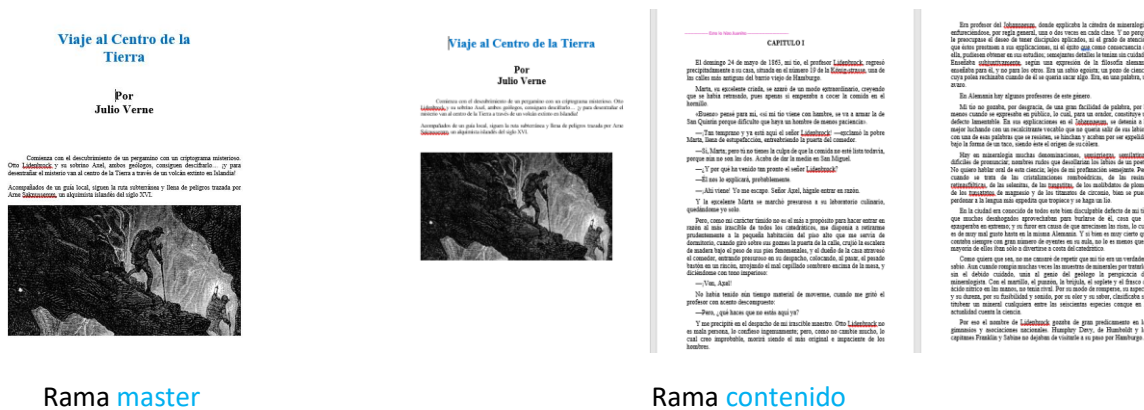
Podemos notar la diferencia en el archivo novela.docx cuando nos cambiamos a la rama principal master.

**Nota:** Cada vez que cambies de rama asegúrate de cerrar el archivo novela.docx

```
Paulina@DESKTOP-9A4451G MINGW64 ~/Documents/practica_git (contenido)
$ git checkout master
Unlink of file 'novela.docx' failed. Should I try again? (y/n) y
Switched to branch 'master'

Paulina@DESKTOP-9A4451G MINGW64 ~/Documents/practica_git (master)
$ |
```

Una vez que te encuentras en master, vuelve abrir tu archivo novela.docx y notarás que solo se encuentra la portada. Recuerda que la información que se agregó esta guardado en la rama contenido.



Ahora crea una nueva rama llamada conclusión, en esta rama pondremos la crítica a este libro.

**Nota:** La rama debe contener la copia de la versión de la rama contenido. Para esto posicónate en la rama contenido con git checkout.

```
Paulina@DESKTOP-9A4451G MINGW64 ~/Documents/practica_git (contenido)
$ git branch conclusion

Paulina@DESKTOP-9A4451G MINGW64 ~/Documents/practica_git (contenido)
$ git checkout conclusion
Switched to branch 'conclusion'

Paulina@DESKTOP-9A4451G MINGW64 ~/Documents/practica_git (conclusion)
$ |
```

Una vez posicionados en esta nueva rama (conclusión), en nuestro archivo Word anexaremos la información referente a la crítica del libro.

un verdadero poliglota.

Al dar con esta dificultad, iba a dejarse llevar de su carácter violento, y ya veía ya venir una escena desagradable, cuando dieron las dos en el reloj de la chimenea.

En aquel mismo momento, abrió Marta la puerta del despacho, diciendo: —La sopa está servida.

—El diablo cargue con la sopa —exclamó furibundo mi tío—, y con la que la ha hecho y con los que se la coman!

Marta se marchó asustada; yo salí detrás de ella, y, sin explicarme cómo, me encontré sentado a la mesa, en mi sitio de costumbre.

Esperé algunos instantes sin que el profesor viniera. Era la primera vez, que yo sepa, que faltaba a la solemnidad de la comida. ¡Y qué comida, Dios mío! Sopas de perejil, tortilla de jamón con aceitunas y nuez moscada, solomillo de ternera con compota de ciruelas, y, de postre, langostinos en dulce, y todo abundantemente regado con exquisito vino del Mosá.

He aquí la apetitosa comida que se perdió mi tío por un viejo papelucho. Yo, a fuer de buen sobrino, me crispé en el deber de comer por los dios, y me amangué de un modo asombroso.

—¿No he visto en los días de mi vida una cosa semejante! —decía la buena Marta, mientras me servía la comida. ¡Es la primera vez que el señor Lidenbrock falta a la mesa!

—No se concibe, es efecto.

—Esto parece presagio de un grave acontecimiento —añadió la vieja criada, sacudiendo sentenciosamente la cabeza.

Pero, a mi modo de ver, aquello lo que presagiaba era un escándalo horrible que iba a promover mi tío tan pronto se percatase de que había devorado su ración.

Me estaba yo comiendo el último langostino, cuando una voz estentórea me hizo volver a la realidad de la vida, y, de un salto, me trasladé del comedor al despacho.

— Esto lo hizo Claudia —

### Mi crítica

Es curioso ver cómo se disfruta o se sufre de la literatura dependiendo de la edad, el estado de ánimo o incluso la época del año. Hace más de veinte años que leí por primera vez este libro. De aquel momento, en mi mente pesa el sentimiento de haber descubierto un mundo nuevo. Pero, ¿qué era tan fascinante de un mundo subterráneo como el que Verne describe? No me pasó a decir la variedad de los hechos, ni tan siquiera a pensar en cómo se pudo documentar tan exhaustivamente en cada detalle de la novela. La disfruté. Únicamente hice eso, y así quedó grabado en mi mente.

Cuando decidimos crear el mapa de los capítulos en el quinto libro, fui el último en elegir autor y novela. Tuji a mis compañeros que la historia antes, más por curiosidad de ver por qué decíamos nos movíamos que por otro motivo. Al ver que ninguno de ellos eligió al bueno de Julio, saqué que este escritor no podía fallar en la lectura. Por un par de días pensé que, con las reminiscencias que aún revolvían en mi mente de la obra tendida bastante para crear la novela, y así dedicar mi escaso tiempo a otras lecturas que me esperaba. Pero al tercer día, como si de una insurrección se tratase, mi pensamiento fue otro. ¡El correo, el que toca. Debía volver a leer la historia del capítulo del profesor Lidenbrock. Porque, aunque el término no estaba acortado, ya es seguro que este seche era un fide de libro, nunca mejor dicho.

Y hoy, al escribir la novela, lo hago con un regusto extraño, a medias entre dulce y amargo. El dolor me llegó de saber que aquella a libro como este continúa leyendo, pues en la época que nací la primera lectura fue la del descubrimiento de las aficiones, la de ir conociendo el mundo y eludir qué lo mudo y qué no. Seguramente si el libro y el escritor hubiesen sido otros, no estaría ahora sí en este blog, si mucho menos en este mundillo literario del que tanto disfruto.

El toque amargo lo imprimió el hecho de darse cuenta que ya no disfruto tanto de la lectura. Ahora me fijo mucho más en los detalles, pienso en cómo se habrá documentado esto. Viene en un trabajador puntual, y el trabajo de documentación de esta novela no tuvo que ser mucho de peso. Solamente hay que ver las insensurables referencias a la profecía, en la forma de poner una trama tan detallada como esta.

Cuando meche acabé y cerré el libro, así que los sentimientos hacia la novela han cambiado. Si para bien, si para mal. Simplemente han cambiado.

Y ahora, porque supongo que han llegado a esta novela para conocer un poquito de la novela, no de un país mental, se constata que es una novela literaria, y más si se dijo que cuando se escribió, allá por 1864, el mundo seguía estando oscuro. Verne optó por contar una historia desde un profesor y su asistente deciden, tras descubrir un centro, viajar hasta Islandia (menciona una resaca aparte la Islandia pobre, trabajadora, alejada del turismo y las aventuras literarias que tan maravillosamente describe Verne) para descender por el cráter de un volcán e ir descubriendo un mundo debajo de nuestros pies.

Guardamos y subimos los cambios a la rama conclusión.

```
Paulina@DESKTOP-9A4451G MINGW64 ~/Documents/practica_git (conclusion)
$ git commit -am "Esta fue la crítica hecha por Claudia"
[conclusion 0643c5f] Esta fue la crítica hecha por Claudia
1 file changed, 0 insertions(+), 0 deletions(-)
```

## Fusionando diferentes versiones

Si te das cuenta tenemos tres ramas: la principal master, contenido y conclusión. Utiliza **git branch** para conocer tus ramas en Git. En cada rama tenemos una versión diferente del archivo novela.docx

```
Paulina@DESKTOP-9A4451G MINGW64 ~/Documents/practica_git (conclusion)
$ git branch
* conclusion
  contenido
  master
```

Ahora queremos fusionar la información para que solo quede un archivo que contenga toda la información que se fue agregando en las ramas. Para hacerlo utilizamos un comando llamado **git merge**. Este comando se ejecuta en la rama donde deseas que la fusión de versiones se guarde.

En esta práctica, primero fusionamos la versión entre crítica y conclusión. Para hacerlo nos posicionamos en la rama de contenido y a continuación ejecutamos **git merge nombre\_de\_la\_rama \_a\_fusionar** y el mensaje.

```
Paulina@DESKTOP-9A4451G MINGW64 ~/Documents/practica_git (contenido)
$ git merge conclusion -m "Fusion entre contenido y critica"
Updating d976c2e..0643c5f
Fast-forward (no commit created; -m option ignored)
 novela.docx | Bin 115816 -> 118880 bytes
 1 file changed, 0 insertions(+), 0 deletions(-)
```

Ahora abre tu archivo novela.docx y observa que el contenido de Mi crítica ya se anexo.

Por último, fusiona la versión de contenido a la rama principal ya que en esta rama solo se tiene la portada, pero nada de contenido y conclusión. En master se crea nuestra versión final.

```
Paulina@DESKTOP-9A4451G MINGW64 ~/Documents/practica_git (contenido)
$ git checkout master
Switched to branch 'master'

Paulina@DESKTOP-9A4451G MINGW64 ~/Documents/practica_git (master)
$ git merge contenido -m "Version final"
Updating 8867d85..0643c5f
Fast-forward (no commit created; -m option ignored)
 novela.docx | Bin 104031 -> 118880 bytes
 1 file changed, 0 insertions(+), 0 deletions(-)
```

- Diferencias entre Git reset vs Git rm
- Para quitar el repositorio creado por git init, hay que primero posicionarse en la dirección de la carpeta y luego utilizar el comando **rm -rf .git**
- Para eliminar una rama de nuestro repositorio local ejecutaremos el siguiente comando  
`git branch -d nombre_rama`
- En el caso de que esa rama contenga trabajos sin fusionar, el comando anterior nos devolverá el siguiente error:

```
error: The branch 'nombre-rama' is not an ancestor of your current HEAD.
If you are sure you want to delete it, run 'git branch -D nombre-rama'.
```

- Si aun así queremos eliminar esa rama, se puede forzar el borrado de la siguiente manera:

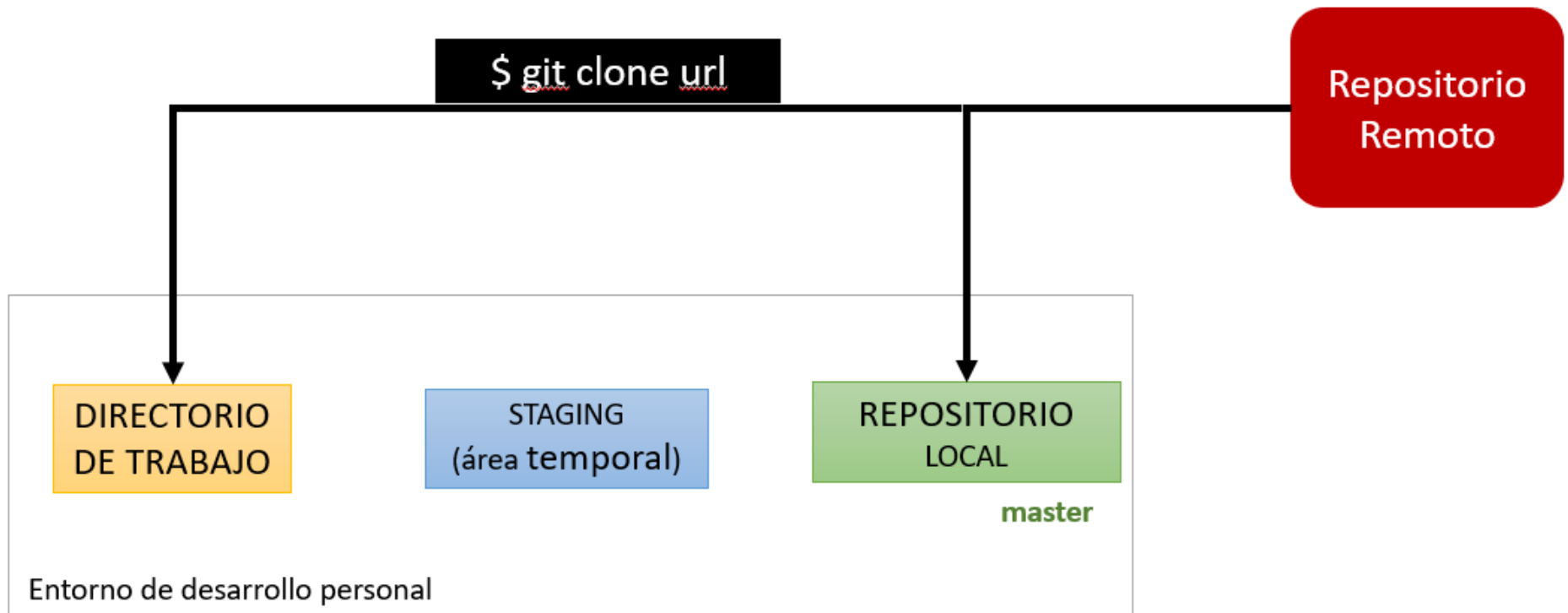
```
$ git branch -D nombre-rama
```

- En el caso de querer eliminar una rama del repositorio remoto, la sintaxis será la siguiente:

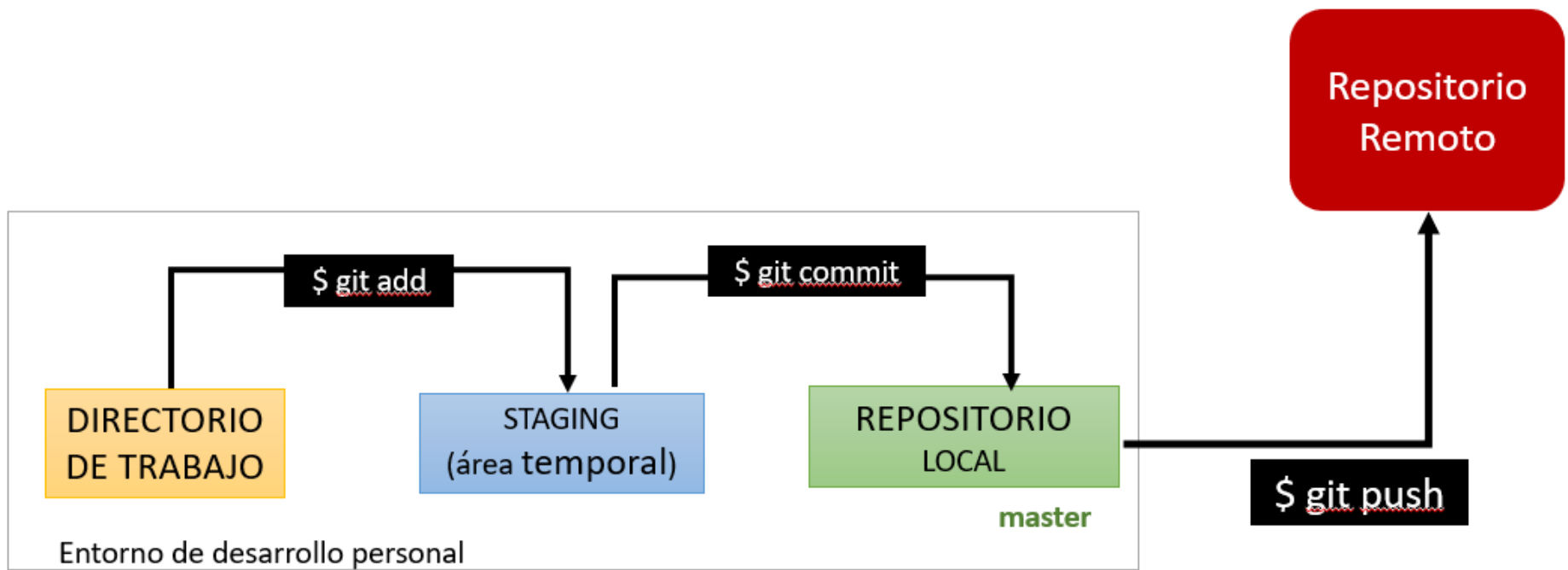
```
$ git push origin :nombre-rama
```



# Ciclo básico de trabajo en GitHub

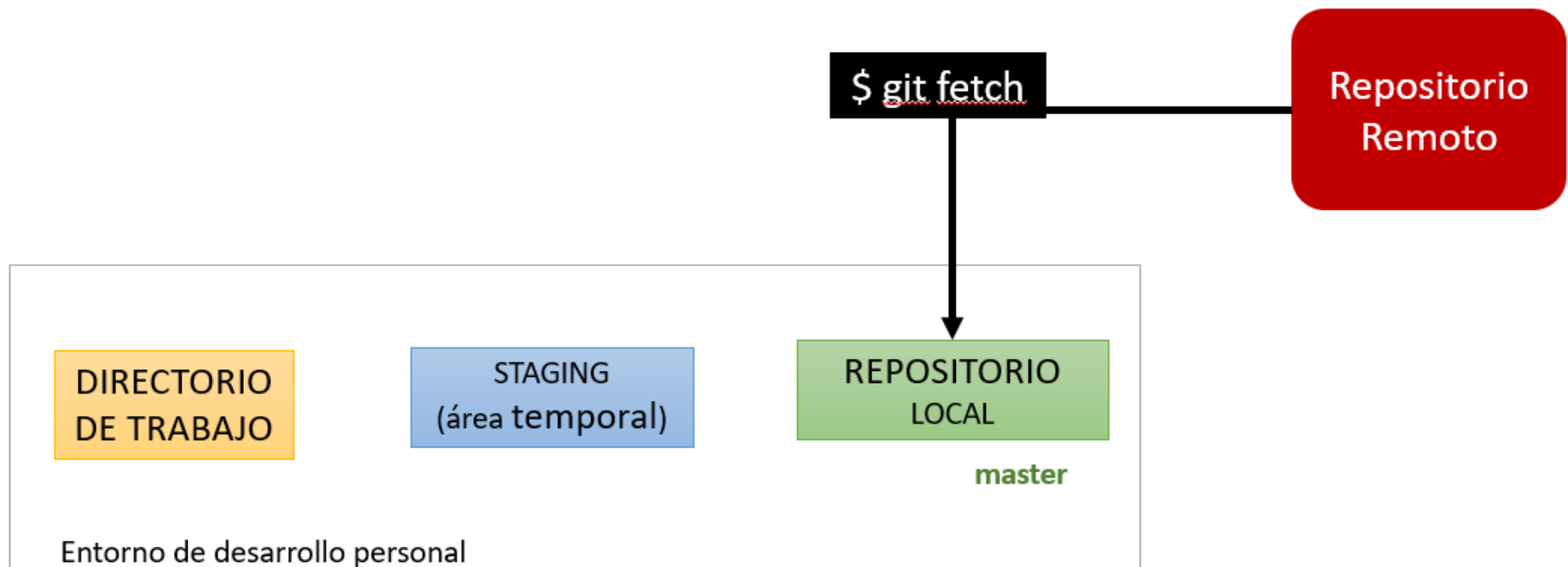


Para traernos los archivos de un servidor remoto, lo que hacemos es clonar.  
Hace todos los cambios históricos en el repositorio y el directorio y deja Staging quieto



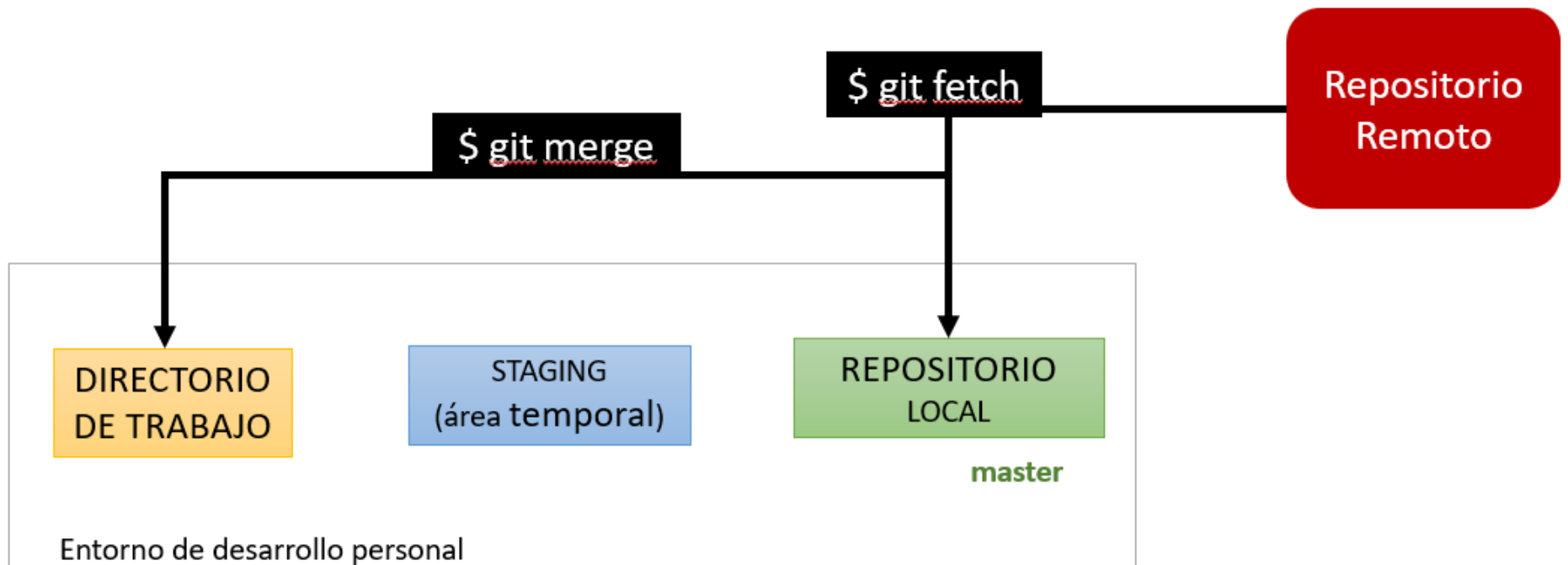
Para pasar la ultima versión de mis archivos locales (HEAD del master) al repositorio remoto



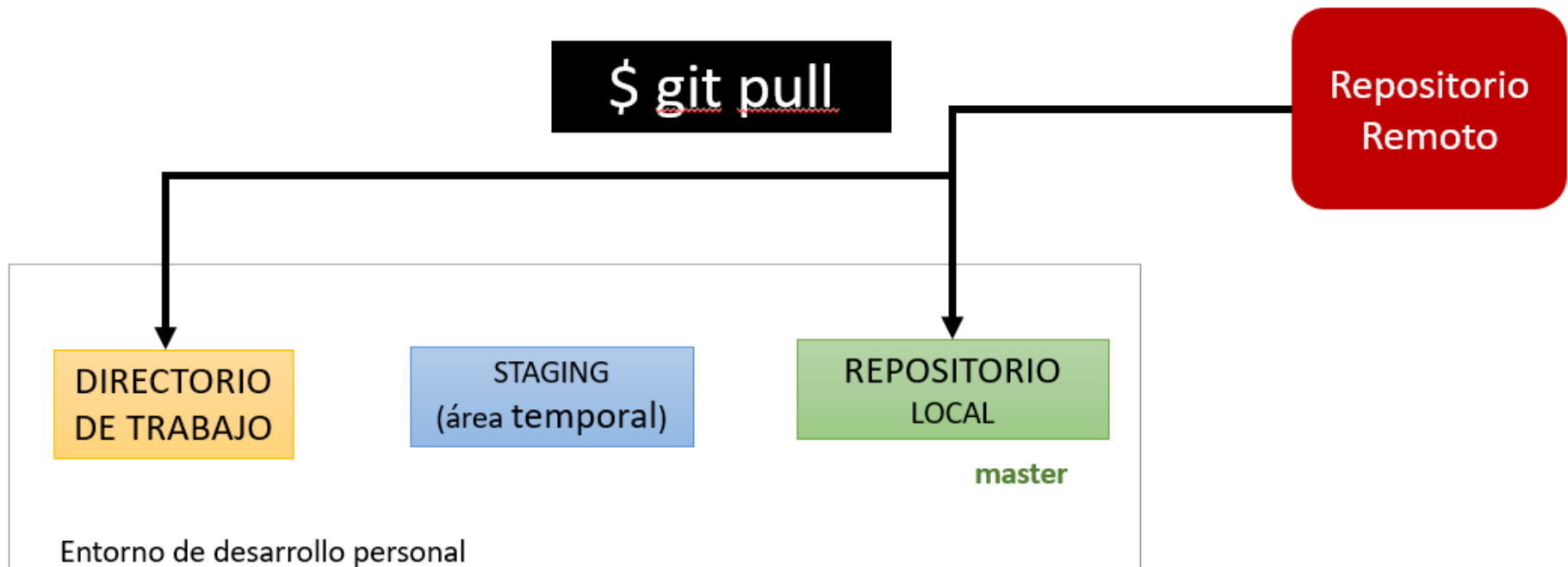


Cuando ya se clono el repositorio remoto al local, pero alguien mas hizo un cambio después y se necesita traer ese cambio al repositorio local.

Nota que `$git fetch` no lo copia en el directorio de trabajo ni en el `staging`.

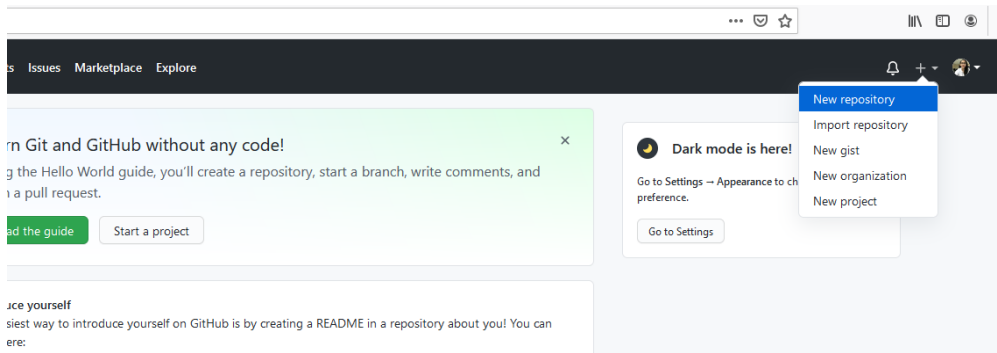


Para que lo copie en el directorio y el staging, se tiene que fusionar la ultima versión que esta en el repositorio local con la versión actual se utiliza \$git merge



Existe un comando que fusiona tanto la instrucción merge como fetch, para que lo que jalemos del repositorio remoto, se guarde tanto en el local como en el directorio de trabajo y es el \$git pull.

Dentro de GitHub ya logeados con su cuenta, creamos un nuevo 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 \*

PaulinaPC6

Repository name \*

Practica GIT

Great repository names are [Your new repository will be created as Practica-GIT.](#) about [ideal-octo-goggles?](#)

Description (optional)

Ánisis del libro

☒ Public

Anyone on the internet can see this repository. You choose who can commit.

☐ Private

You choose who can see and commit to this repository.

Initialize this repository with:

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

☐ Add a README file

This is where you can write a long description for your project. [Learn more.](#)

☐ Add .gitignore

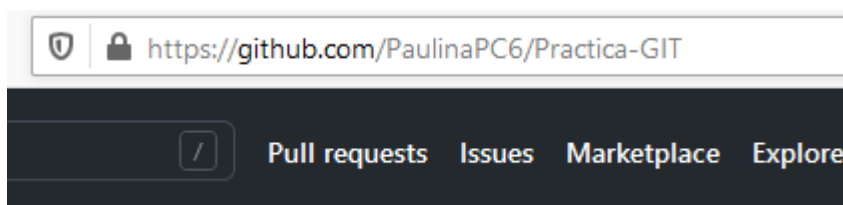
Choose which files not to track from a list of templates. [Learn more.](#)

☐ Choose a license

A license tells others what they can and can't do with your code. [Learn more.](#)

Create repository

Una vez creado el repositorio nos dará una dirección URL al que podremos acceder directamente.



Para agregar a este repositorio el archivo novela.docx que se genero con Git en nuestro local, primero copiamos la URL, regresemos a nuestra consola Git y posicionémonos en la dirección de la carpeta que contiene el archivo novela.docx en la rama master.

Le vamos a decir a Git que vamos agregar un origen remoto de nuestros archivos, utilizando **git remote add origin** y la dirección https.

```
Paulina@DESKTOP-9A4451G MINGW64 ~/Documents/practica_git (master)
$ git remote add origin https://github.com/PaulinaPC6/Practica-GIT
```

A simple vista parece que no paso nada, pero si utilizamos **git remote** veremos que existe *origin* como acceso remoto.

```
Paulina@DESKTOP-9A4451G MINGW64 ~/Documents/practica_git (master)
$ git remote
origin
```


Si ejecutamos git remote -v, nos dice que tenemos un acceso remoto para hacer fetch (traer) y push (enviar)

```
Paulina@DESKTOP-9A4451G MINGW64 ~/Documents/practica_git (master)
$ git remote -v
origin https://github.com/PaulinaPC6/novela.git (fetch)
origin https://github.com/PaulinaPC6/novela.git (push)
```

Ya que queremos enviar nuestro archivo novela.docx al repositorio remoto creado en GitHub, utilizamos git push.

```
Paulina@DESKTOP-9A4451G MINGW64 ~/Documents/practica_git (master)
$ git push origin master
Enumerating objects: 15, done.
Counting objects: 100% (15/15), done.
Delta compression using up to 8 threads
Compressing objects: 100% (10/10), done.
Writing objects: 100% (15/15), 148.03 KiB | 9.25 MiB/s, done.
Total 15 (delta 4), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (4/4), done.
To https://github.com/PaulinaPC6/Practica-GIT
 * [new branch]      master -> master
```


Si regresas a GitHub y recargas la página notarás que ya se ha agregado el archivo novela.docx en su última versión ya que es la que existía en la rama master y fue la que subimos al repositorio de GitHub.


 PaulinaPC6 / Practica-GIT

[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#) [Settings](#)

master 1 branch 0 tags

Go to file Add file Code

 PaulinaPC6 Version final 75cd03b 20 minutes ago 5 commits

 novela.docx Version final 20 minutes ago

Help people interested in this repository understand your project by adding a README. [Add a README](#)

## Configurar múltiples colaboradores en un repositorio de GitHub

Por defecto, cualquier persona puede clonar o descargar tu proyecto desde GitHub pero no pueden crear commits, ramas, etc. Es necesario añadir a cada persona de nuestro equipo como colaborador de nuestro repositorio.

Para esto, debemos entrar a la configuración de colaboradores de nuestro proyecto (Repositorio > Settings > Colaboradores y añadir el email o username de los nuevos colaboradores.

### Agregando otros colaboradores al repositorio de GitHub

Por ejemplo, el repositorio en GitHub donde se agregó la versión final de novela.docx, otro usuario podría visitar ese repositorio y clonarlo en el suyo utilizando la URL del repositorio.

El colaborador necesita irse a su Git local logearse con su username y su correo, después posicionarse en la carpeta en la que va a trabajar y utilizar **git clone** más la URL.

```
Paulina@DESKTOP-9A4451G MINGW64 ~/Documents/Alberto
$ git clone https://github.com/PaulinaPC6/Practica-GIT
Cloning into 'Practica-GIT'...
remote: Enumerating objects: 15, done.
remote: Counting objects: 100% (15/15), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 15 (delta 4), reused 15 (delta 4), pack-reused 0
Receiving objects: 100% (15/15), 148.03 KiB | 924.00 KiB/s, done.
Resolving deltas: 100% (4/4), done.
```

Como el repositorio original es público, al momento de clonar nunca pidió usuario y contraseña.

*Nota: Todos los repositorios públicos están al alcance de cualquier usuario y con permisos de clonar toda su información.*

Si el colaborador abre la carpeta donde está trabajando, se dará cuenta que ya existe el archivo .docx en su computador.

Ahora el nuevo colaborador agrega contenido al archivo de la novela.docx subió sus cambios en su repositorio local.

```
Paulina@DESKTOP-9A4451G MINGW64 ~/Documents/Alberto/Practica-GIT (master)
$ git add novela.docx

Paulina@DESKTOP-9A4451G MINGW64 ~/Documents/Alberto/Practica-GIT (master)
$ git commit -m "Alberto agrego capitulos"
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
```

Podemos conocer todo el historial de commits que se han hecho a ese archivo trayendo todo el repertorio de GitHub.

```
Paulina@DESKTOP-9A4451G MINGW64 ~/Documents/Alberto/Practica-GIT (master)
$ git pull origin master
From https://github.com/PaulinaPC6/Practica-GIT
* branch      master      -> FETCH_HEAD
Already up to date.
```

Con **git log** conocemos el historial

```

Paulina@DESKTOP-9A4451G MINGW64 ~/Documents/Alberto/Practica-GIT (master)
$ git log
commit 75cd03b04c520da18491cee25695d24596adc2d9 (HEAD -> master, origin/master,
origin/HEAD)
Author: PaulinaPC <paulina.geopc@gmail.com>
Date: Sun May 16 17:06:45 2021 -0500

    Version final

commit 0643c5f822967447ef1da640da01bb4ce68129ca
Author: PaulinaPC <paulina.geopc@gmail.com>
Date: Sun May 16 13:18:35 2021 -0500

    Esta fue la crítica hecha por Claudia

commit d976c2e86b682a9c830aa8a0ed19fb3ca9cdd219
Author: PaulinaPC <paulina.geopc@gmail.com>
Date: Sun May 16 13:03:25 2021 -0500

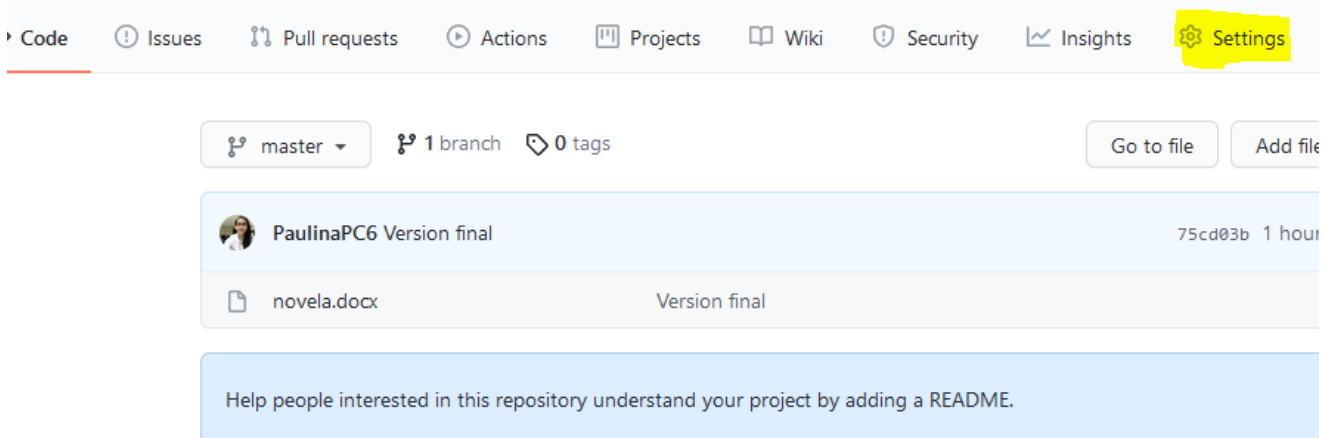
    Este es el archivo con capitulo 1 y 2 de Juanito

commit 8867d855c598d2bc958746e22281a3c2c62ef28f
Author: PaulinaPC <paulina.geopc@gmail.com>

```

Las modificaciones que hizo el nuevo colaborador se encuentran solamente en el local de este usuario mas no en el repositorio remoto. Para enviar estos cambios al repositorio, primero el usuario que creo el repositorio remoto en GitHub tiene que dar los permisos y agregar como colaborador de ese repositorio.

En Git Hub se da clic en la opción de Ajustes que pertenecen al repositorio a compartir [PaulinaPC6 / Practica-GIT](#)



Se abrirá una serie de opciones, en Manage Access se puede invitar a colaborar a otro usuario.

Options
Manage access
Security & analysis
Branches
Webhooks
Notifications
Integrations
Deploy keys
Actions
Environments
Secrets

## Who has access

PUBLIC REPOSITORY


This repository is public and visible to anyone.

Manage

DIRECT ACCESS

0 collaborators have access to this repository. Only you can contribute to this repository.


## Manage access




You haven't invited any collaborators yet


Invite a collaborator

Para agregar a nuevos colaboradores, se pueden buscar por nombre de usuario o email.



Invite a collaborator to Practica-GIT





Add aporras@centrogeo.edu.mx to this repository

El nuevo colaborador le llegara la invitación de colaborar en este repositorio, lo que tiene que hacer es aceptar la invitación y ahora si puede subir sus cambios al repositorio.

**Nota:** Puede pasar que, para conectarte al repositorio, automáticamente en la consola te pide tu nombre de usuario y contraseña. No olvides crear tu acceso remoto de origin al a URL del repositorio colaborativo.