Guia

January 13, 2026

1 Curso Git y GitHub

1.1 ¿Qué es Git?

- Es un **software de control de versiones** que registra los cambios realizados sobre un archivo o conjunto de archivos a lo largo del tiempo.
- Git nos permite **gestionar distintas versiones** de un mismo archivo, pudiendo volver a una versión anterior o una mas reciente cuando sea necesario.

1.2 ¿Qué es un repositorio?

- Cuando trabajamos con GIT lo que creamos es un repositorio, es decir, un lugar en donde guardamos y administramos nuestros archivos.
- Existen dos tipos de repositorios
  - **Locales:** Son aquellos que se crean de forma "local" en nuestra PC, y que no necesariamente son compartidos.
  - **Remotos:** Son aquellos que se encuentran alojados en algún servidor externo y que puede ser accedido desde cualquier lugar. Un ejemplo de repositorio remoto puede ser Git-HUB.

1.3 Inicializar un repositorio

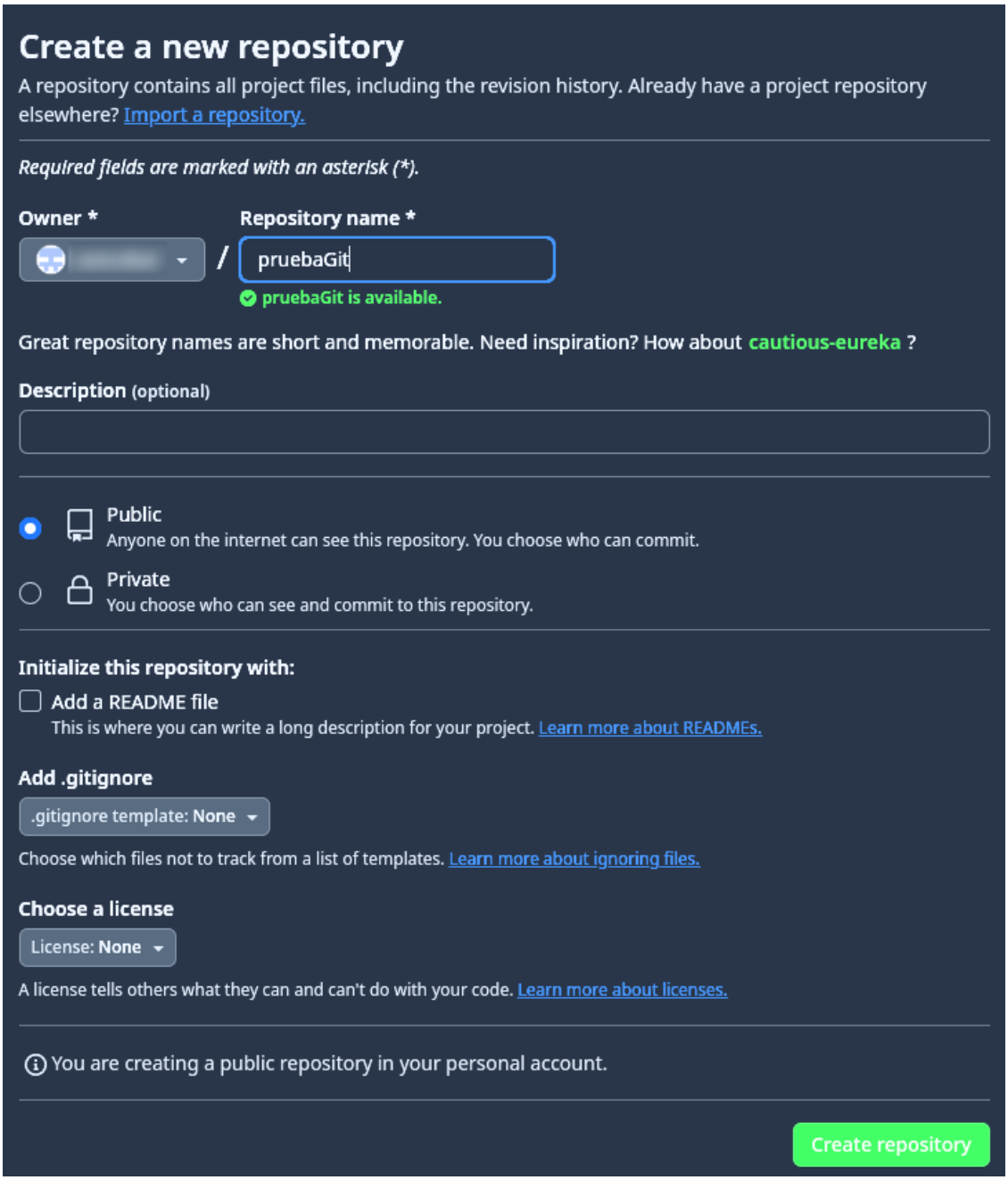
Para iniciar un repositorio dentro de una carpeta, debemos usar el comando `git init` para crear un repositorio local vacío.

```
10:49 $ mkdir pruebaGit
10:53 $ cd pruebaGit/
10:53 $ git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
```

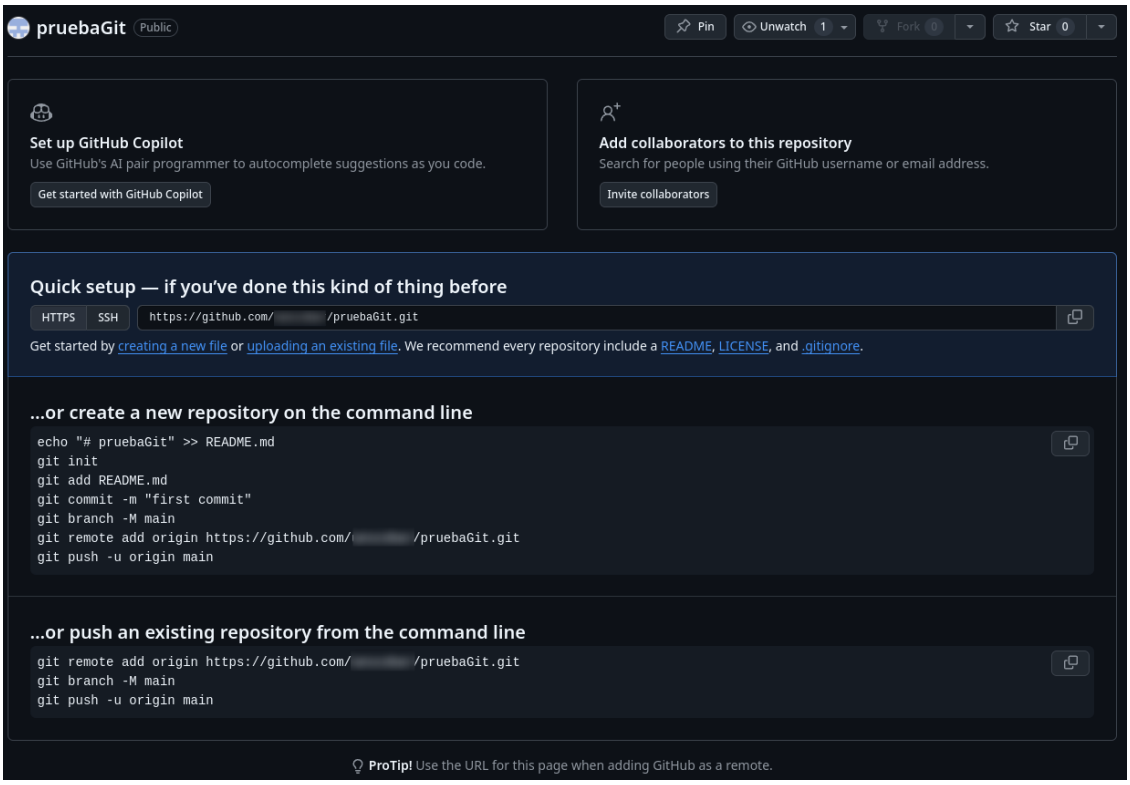
```
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /home/user/pruebaGit/.git/
10:55 $
```

1.4 Repositorio remoto

- Crear una cuenta en github.com, se requiere una cuenta de correo y de preferencia activar el método de autenticación de dos fases usando **authenticator**., elegir un username.
- Desde la página de github es posible crear un repositorio:
  - escoger el nombre, por ejemplo **pruebaGit**
  - escribir una descripción
  - elegir si será público o privado.
  - opcionar elegir un tipo de licencia (considerar como opcion MIT License)
  - y precionar el botón de crear repositorio.



Despues de crear el repositorio remoto, nos mostrará una pantalla con la información necesaria para trabajar con el repositorio.



En los pasos anteriores, el repositorio creado en se llama **pruebaGit**, para poder enlazarlo con el repositorio local creado anteriormente.

Ahora, vamos a configurarlos para que se enlacen.

Dentro del repositorio local, se va a contar con una carpeta llamada **.git**, en la cual se va a almacenar la información de todos las versiones que se vayan trabajando del código, así como las ramas (branchs). La presencia de este directorio indica que el repositorio está correctamente inicializado.

Antes de conectar con el repositorio remoto, tenemos que identificarnos, tenemos que indicar de quién es este repositorio mediante los siguientes comandos.

`git config user.name "nombreUsuarioGithub"` `git config user.email "emailRegistradoGithub"`

Ahora se sabe que este repositorio es del usuario con email que se indica.

Ahora para comunicar el repositorio local con el repositorio remoto se usa el siguiente comando

`git remote add origin "https://github.com/nombreUsuarioGithub/pruebaGit.git"`

Una vez hecho esto, ya va a quedar configurado nuestro repositorio local con el repositorio remoto.

```
10:55 $ ls -a
.  ..  .git
11:44 $ git config user.name "nombreUsuarioGithub"
11:44 $ git config user.email "emailRegistradoGithub"
```

```
11:44 $ git remote add origin "https://github.com/nombreUsuarioGithub/pruebaGit.git"
```

1.5 git add + git commit + git push

Para pasar archivos de nuestro repositorio local al repositorio remoto.

Para realizar la prueba, vamos a crear un archivo en el repositorio local cuyo contenido será

hola.txt

```
No te olvides de que manejar git es esencial porque es donde vas a evidenciar lo que
has creado, es tu portafolio profesional.
```

```
11:45 $ vi hola.txt
11:52 $ cat hola.txt
No te olvides de que manejar git es esencial porque es dónde vas a evidenciar lo que
has creado, es tu portafolio profesional.
```

Ahora, vamos a conocer el estado en el que se encuentra el estado del repositorio con el comando `git status`

```
11:53 $ git status
On branch master

No commits yet

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

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

Allí aparece un archivo que se encuentra fuera del stage, es decir, que todavía no ha sido ”comiteado”.

Ahora, necesitamos indicar que deseamos agregar el archivo al repositorio con el comando `git add hola.txt`

```
11:54 $ git add hola.txt
```

Ahora verificamos el status del repositorio nuevamente con el comando `git status`

```
11:57 $ git status
On branch master

No commits yet

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

11:58 $
```

Es decir que el archivo ya fue agregado de forma ”temporal”, es decir, solo está considerado para ser agregado y por el momento no se ha cargado en el repositorio remoto. Una forma de verlo es considerar es que el archivo esta en el repositorio local y que se ha marcado como algo que en un futuro debe ser cargado en el repositorio remoto, y aún podemos volver atrás, es decir, dejarlo de considerar, **pero por el momento si queremos que sea parte de lo que pasaremos al repositorio remoto**.

Para confirmar y considerar de forma definitiva que el archivo **hola.txt** debe ser parte de la nueva versión del repositorio remoto, lo hacemos mediante un commit, que es el comando `git commit -m "mensaje"`

```
11:58 $ git commit -m "en este commit se hicieron cambios commit 1"
[master (root-commit) d692d0f] en este commit se hicieron cambios commit 1
 1 file changed, 1 insertion(+)
 create mode 100644 hola.txt
```

Para corroborar que el commit se llevó a cabo se usa `git log`

```
12:06 $ git log
commit d692d0fa4732bf24c3f47d3149ac5d0d0077ec0e (HEAD -> master)
Author: nombreUsuarioGithub <emailRegistradoGithub>
Date:   Wed Jan 22 12:06:05 2025 -0600

    en este commit se hicieron cambios commit 1
```

Ahora, para poder pasar este archivo al repositorio remoto, debemos usar el comando `git push -u [nombre de la rama]`

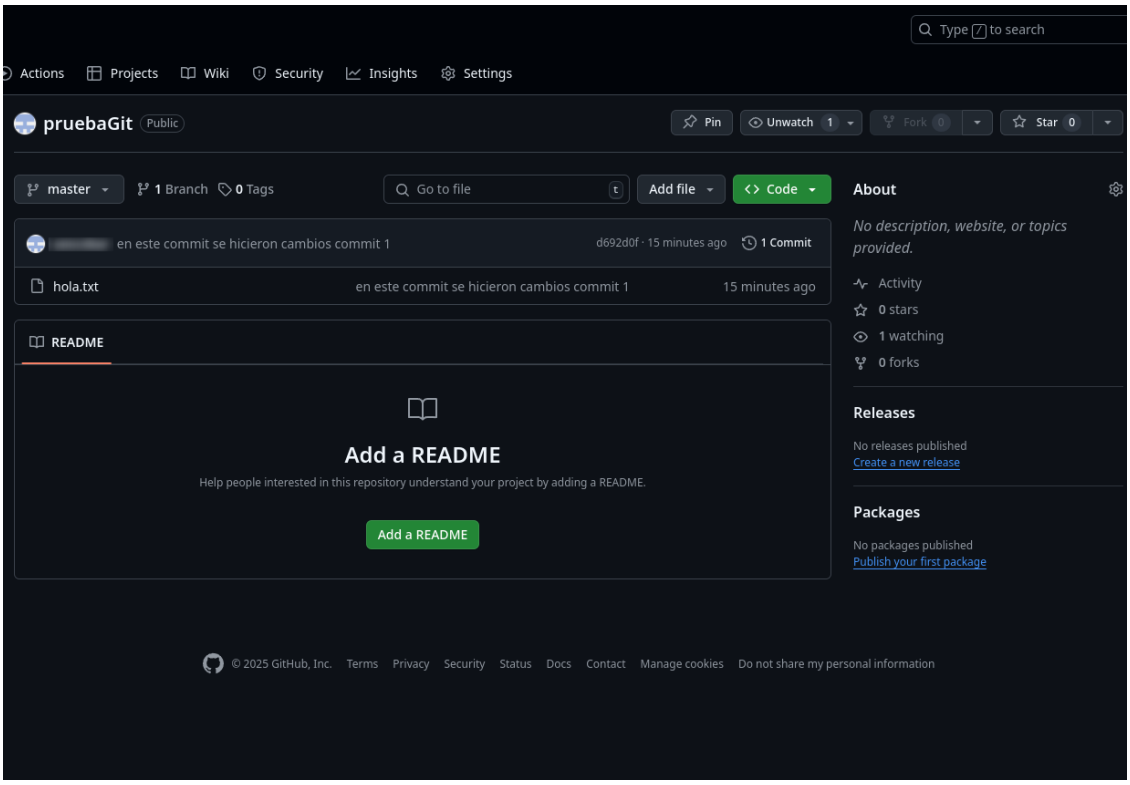
Una rama es una linea de trabajo distinta a la principal que tengamos. Por ejemplo, si consideramos los coches, hay un solo modelo, pero con distintas versiones que contienen distintas caracteristicas. Cuando nos decidimos cual es la versión que nos interesa trabajar, esa versión es la principal (master). De la misma forma trabaja git, y siempre estas ramas van a quedar o desembocar en una rama principal.

Como no hemos creado ramas, git creó una rama denominada 'master' o 'main' y es la línea de trabajo principal, de allí van a partir las líneas de trabajo extras, pero se pueden crear ramas adicionales. Por lo pronto usaremos el comando `git push -u origin master`, recoradar que el repositorio remoto lo denominamos origin.

Una vez ejecutado el comando nos va a pedir que nos autentiquemos con el usuario y token de acceso de github.

```
12:08 $ git push -u origin master
fatal: cannot exec '/usr/bin/ksshaskpass': No such file or directory
Username for 'https://github.com': nombreUsuarioGithub
fatal: cannot exec '/usr/bin/ksshaskpass': No such file or directory
Password for 'https://nombreUsuarioGithub@github.com':
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 6 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 319 bytes | 319.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/nombreUsuarioGithub/pruebaGit.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.
```

Ahora la rama master ya ha sido "trakeada" al repositorio remoto, para corroborarlo, podemos ir al navegador y en la dirección del repositorio podemos comprobar que ya esta allí el archivo.



1.6 clone pull

el comando se usa para copiar todo un repositorio y traerlo al equipo local, para eso se usa el comando `git clone [url del repositorio remoto]`

```
12:34 $ git clone https://github.com/nombreUsuarioGithub/pruebaGit
Cloning into 'pruebaGit'...
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 (from 0)
Receiving objects: 100% (3/3), done.
```

Ahora, cuando por ejemplo, trabajamos en conjunto con otras personas, y esas personas hacen cambios en el repositorio, es necesario actualizar la información que tenemos del repositorio sobre la rama que nos hayan indicado, lo cual se hace con el comando `git pull`

Por ejemplo, ahora desde el sitio de github creemos un nuevo archivo dentro del repositorio que estamos trabajando, para lo cual **dar clic en el botón "Add file"**, el archivo se llama **pruebita**, se creará dentro de la rama ***master***.

pruebita

```
hola soy una prueba para el PULL
```

Ahora, **dar click en el botón "Commit changes"**, se abre la ventana y solo hay que dar clic en el botón de "Commit changes", se puede editar la información requerida del commit en caso de ser necesario. Este archivo únicamente se ha creado en el repositorio remoto.

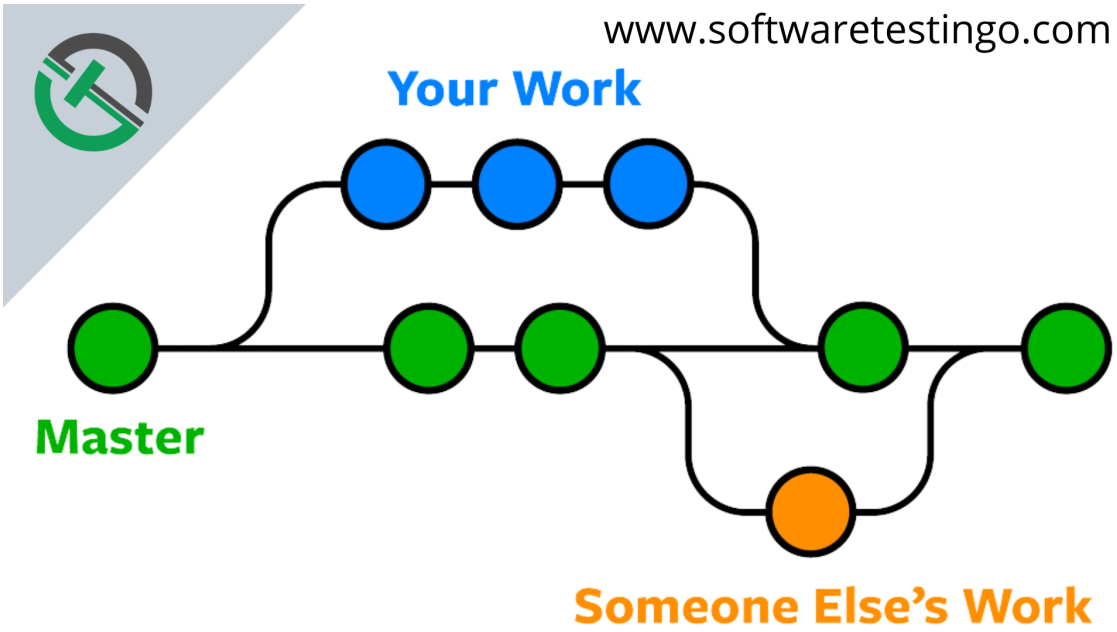
Lo que interesa es que los cambios del repositorio remoto sean traidos al repositorio local, en dónde estamos trabajando. Para hacer eso, desde dentro del directorio del repositorio, recordar que el apodo que le pusimos al repositorio es origin y la rama es master, por lo que ejecutamos el comando `git pull origin master`

```
12:44 $ cd pruebaGit/
12:45 $ pwd
/home/user/pruebaGit
12:45 $ git pull origin master
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 951 bytes | 475.00 KiB/s, done.
From https://github.com/nombreUsuarioGithub/pruebaGit
 * branch            master     -> FETCH_HEAD
   d692d0f..08130c5  master     -> origin/master
Updating d692d0f..08130c5
Fast-forward
 pruebita | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 pruebita
12:49 $
```

1.7 ¿Qué es una rama?

En git se pueden manejar varias versiones de un mismo proyecto, a eso se le denomina ramas, y es posible estar cambiando entre una y otra todo el tiemp, únicamente intercambiando entre las ramas.

- Las ramas pueden definirse como "punteros" para cada uno de los cambios o versiones que queremos armar de nuestro proyecto de desarrollo.
- Cada rama representa una línea independiente de desarrollo.
- Git cuenta con una rama principal master/main de la cual parten todas las demás ramas que se pueden crear.



Podemos trabajar con varias personas al mismo tiempo y en donde cada uno crea su propia rama y luego las ramas se unen para tener una versión final.

1.7.1 Crear ramas

Primero hay que ubicarse dentro del directorio del repositorio local y usar el comando `git branch` y nos indica que estamos en la rama **master**.

```
12:49 $ git branch
* master
13:55 $
```

Ahora crearemos la rama **rama1** con el comando `git branch rama1`

```
13:55 $ git branch rama1
13:56 $
```

Se ha creado la **rama1**, verificamos nuevamente en qué rama estamos `git branch`, como podemos ver estamos en la rama **master**

```
13:56 $ git branch
* master
  rama1
```

```
13:58 $
```

1.7.2 Cambiar de nombre a las ramas

Los nombres de las ramas se pueden editar, por ejemplo, cambiemos el nombre de **rama1** a **ramita1**.

```
git branch -m rama1 ramita1
```

```
13:58 $ git branch -m rama1 ramita1
14:00 $
```

Verificamos que se haya cambiado el nombre con `git branch`

```
14:00 $ git branch
* master
  ramita1
14:00 $
```

1.7.3 Cambiar de rama

Para cambiar de rama, usamos el siguiente comando `git checkout [rama]`.

```
git checkout ramita1
```

```
14:00 $ git checkout ramita1
Switched to branch 'ramita1'
```

Verificamos ahora la rama en la que estamos ubicados. `git branch`

```
14:04 $ git branch
  master
* ramita1
```

1.7.4 Eliminar rama

A veces deseamos eliminar la rama por alguna razon. `git branch -d [rama]`

```
git branch -d ramita1
```

```
14:06 $ git branch -d ramita1
error: cannot delete branch 'ramita1' used by worktree at '/home/user/pruebaGit'
14:08 $
```

No se puede borrar en la rama mientras se está en uso. Primero hay que cambiarse a la rama **master** `git checkout master`

```
14:08 $ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
```

Y ahora si se procede a eliminar la rama **ramita1**

```
git branch -d ramita1
```

```
14:10 $ git branch -d ramita1
Deleted branch ramita1 (was 08130c5).
```

Ahora se verifica que la rama **ramita1** ya no existe.

```
git branch
```

```
14:11 $ git branch
* master
```

1.8 Crear archivos en distintas ramas

Ejemplo para crear archivos en distintas ramas

```
git branch rama1
```

```
git branch
```

```
git checkout rama1
```

```
git branch
```

```
14:12 $ git branch rama1
14:15 $ git branch
* master
  rama1
14:15 $ git checkout rama1
Switched to branch 'rama1'
14:15 $ git branch
  master
* rama1
14:15 $
```

Ahora vamos a crear archivos que no van a estar en **master**

`touch texto.txt touch texto2.txt`

Se crean dos archivos, pero no se indicó que estos archivos sean exclusivos de la **rama1**, por lo que al cambiar de rama a la rama **master**, estos archivos aparecen en esa rama.

`git checkout master ls`

```
14:17 $ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
14:18 $ ls
hola.txt  pruebita  texto2.txt  texto.txt
```

Podemos observar que los archivos también se encuentran en la rama **master**

Ahora nos regresamos a la rama **rama1**

`git checkout rama1`

Y ahora lo que debemos hacer es agregar y hacer commit para indicar que estos cambios son exclusivos de la **rama1**

`git add .` este comando agrega todos los archivos que estamos trabajando en esta rama `git commit -m "estamos haciendo commit en los archivos"`

```
14:21 $ git add .
14:22 $ git commit -m "estamos haciendo commit en los archivos"
[rama1 7099393] estamos haciendo commit en los archivos
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 texto.txt
 create mode 100644 texto2.txt
```

si damos el comando `ls` dentro de la rama **rama1**

```
14:23 $ ls
hola.txt  pruebita  texto2.txt  texto.txt
```

Y si ahora nos salimos de la rama **rama1** y nos vamos a la rama **master**

`git checkout master`

```
14:25 $ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
```

---

y damos el comando `ls`

```
14:25 $ ls
hola.txt  pruebita
```

Se puede observar que los archivos **texto.txt** y **texto2.txt** ya no se encuentran, porque estamos volviendo a una versión anterior que es **master** y en esa versión anterior aún no se han agregado.

Y si cambiamos a la **rama1**, ahora si vemos los archivos. `git checkout rama1 ls` ***

```
14:27 $ git checkout rama1
Switched to branch 'rama1'
14:29 $ ls
hola.txt  pruebita  texto2.txt  texto.txt
14:30 $
```

De esta forma, podríamos realizar cambios sin afectar la rama principal, así mismo, cuando la funcionalidad está completa o ya estamos seguros de querer realizar los cambios permanentemente en la rama **master** tenemos que usar otro comando llamado **merge**.

1.9 diff y merge trabajando con ramas

ahora estamos posicionados en la **rama1**, en esta rama estan los dos archivos **texto.txt** y **texto2.txt*.

Usualmente debemos trabajar con múltiples ramas, por lo que podemos ocupar un comando **diff** para conocer las diferencias entre las ramas.

`git diff master rama1`

```
14:30 $ git diff master rama1
diff --git a/texto.txt b/texto.txt
new file mode 100644
index 0000000..e69de29
diff --git a/texto2.txt b/texto2.txt
new file mode 100644
index 0000000..e69de29
```

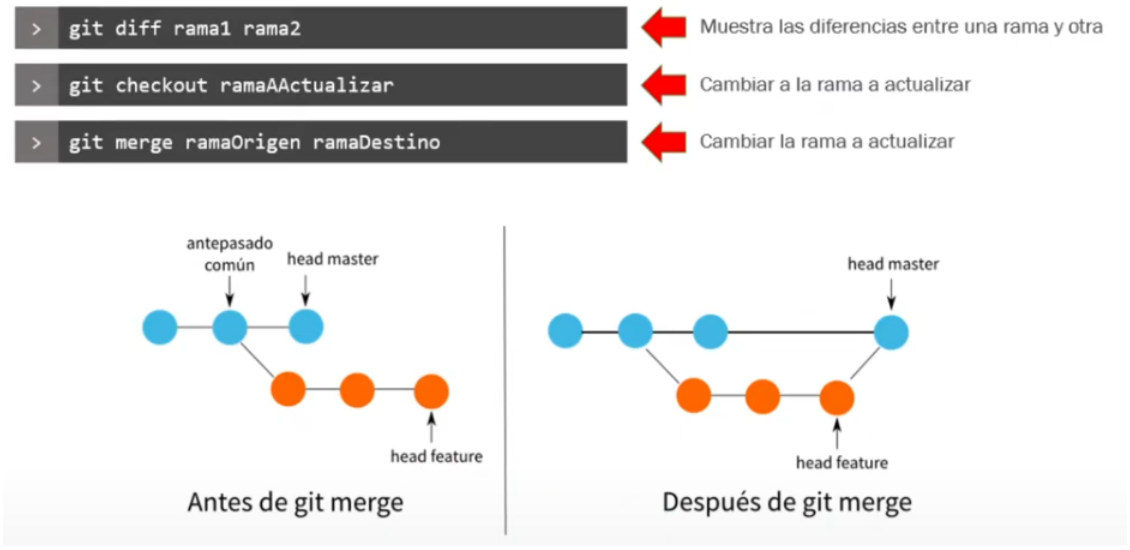
Si cambiamos el orden de las argumentos

```
git diff rama1 master
```

```
14:42 $ git diff rama1 master
diff --git a/texto.txt b/texto.txt
deleted file mode 100644
index e69de29..0000000
diff --git a/texto2.txt b/texto2.txt
deleted file mode 100644
index e69de29..0000000
```

Aqui hay que interpretar **master** con respecto de **rama1** (segundo parámetro con respecti del primer parámetro)

1.9.1 unificar las ramas



En nuestro ejemplo, se hace la prueba con el comando

```
git merge rama1 master
```

```
14:43 $ git merge rama1 master
Already up to date.
```

Muestra que no hay ningun cambio para hacer, y este resultado se produce porque estamos ubicados en la rama que **está más actualizada** entonces, nos debemos parar en la rama que va a recibir los cambios

```
git checkout master
```

```
15:05 $ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
```

---

y ahora si realizamos el comando de **merge**

```
git merge rama1 master
```

```
15:07 $ git merge rama1 master
Updating 08130c5..7099393
Fast-forward
 texto.txt  | 0
 texto2.txt | 0
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 texto.txt
 create mode 100644 texto2.txt
```

Ahora, ya se agregaron los cambios de **rama1** a **master**, sin embargo falta hacer el commit para que se guarden en **master**

```
git add . git commit -m "cambios master"
```

```
15:08 $ git add .
15:11 $ git commit -m "cambios master"
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```

Lo unico que resta es subir los cambios a el repositorio remoto

`git push -u origin master` ***

```
15:13 $ git push -u origin master
fatal: cannot exec '/usr/bin/ksshaskpass': No such file or directory
Username for 'https://github.com': nombreUsuarioGithub
fatal: cannot exec '/usr/bin/ksshaskpass': No such file or directory
Password for 'https://nombreUsuarioGithub@github.com':
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 6 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 325 bytes | 325.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/nombreUsuarioGithub/pruebaGit.git
   08130c5..7099393  master -> master
branch 'master' set up to track 'origin/master'.
```
