

Implantación de Aplicaciones web

1.1 Git y GitHub

IES Las Fuentezuelas

Raúl Yeste Fernández • 02/10/2023

Índice

1.-Introducción

2.- Qué es Git y para qué sirve

3.-Qué es y cómo funciona GitHub

4.-Términos básicos

5.-Prerrequisitos

6.-Flujo de trabajo de Git

6.1-Comandos básicos de Git

7.-Cómo crear un repositorio en GitHub

8.-Instalación de Git

9.-Configuración de Git

10.-Cómo crear repositorios desde cero y almacenar cambios en nuestro repositorio local

10.1-Git init

10.2-.Gitignore

10.3-.Git add

10.4-.Git rm

10.5-.Git commit

10.6-.Git remote conexión repositorio local con uno remoto

10.7-.Git mv

10.8-.Git clone

10.9-.Repositorio remoto compartido

10.10-.Git Branch

10.10.1-.¿Qué es una rama de Git?

10.11-.Git push

10.12-.Git pull

10.13-.Git merge

10.14-.Eliminar un branch

Fin

1-.Introducción

El sistema de control de versiones (VCS) más populares son Git junto con GitHub, sirve para los repositorios y varias herramientas para trabajar con ellos.



2.- Qué es Git y para qué sirve

Git es un Sistema de Control de Versiones Distribuido (DVCS) se utiliza para que cualquier versión sea recuperable cuando se desee .

Git también facilita el registro y comparación de un archivo. Es decir podemos ver todos los cambios que se ha realizado en nuestro ordenador o en cualquier archivo.

Git se usa para:

- El manejo de repositorios y ramas.
- Poder trabajar en equipo.
- Volver a un archivo en el momento en el que estaba correcto.

3-.Qué es y cómo funciona GitHub

GitHub proporciona un servidor Git y un montón de herramientas muy útiles unos repositorios de git personales o de equipo, cómo informar problemas de código, herramientas, revisión, etc.



4-.Términos básicos

GIT→Es un sistema de control de versiones: software que permite registrar versiones
Cambiar el historial del proyecto.

Repositorio→ es cualquier proyecto que GIT esté rastreando y ya tiene un historial
Un GIT que registra sus cambios.

Confirmar → son todos los cambios registrados en el historial de GIT. Cada
Los desarrolladores envían confirmaciones de sus cambios.

Ramas→Son nuevos caminos que toma un proyecto. La rama principal se llama
La maestría es donde el proyecto entra en producción.

Clonar Es una copia exacta del repositorio. Cuando un programador se une a un
trabajo en equipo lo primero que debes hacer es clonar el repositorio en tu máquina local.
Por tanto, cada miembro del equipo tiene un clon del repositorio en su máquina local.

5-.Prerrequisitos

Para usar Git y GitHub, necesitas:

- Un ordenador con Git instalado.
- Una herramienta para usar Git,puedes usar un cliente Git con GUI o simplemente usar una ventana de la terminal.
- Una cuenta de GitHub.

6-.Flujo de trabajo de Git

Estados principales de los archivos en Git:

- Confirmado: Los datos se almacenan de forma segura en su base de datos local.
- Modificado: has realizado cambios en el archivo, pero aún no los has confirmado.
- Provisional: ha marcado un archivo modificado para incluirlo en la siguiente confirmación.

Git y GitHub

- Trabajar localmente: Los cambios locales se almacenan en el directorio de trabajo. Los cambios pasan por el área de ensayo antes de confirmarse en el repositorio local.
- Inserción de cambios: puedes insertar los cambios desde el repositorio local al repositorio remoto en GitHub.

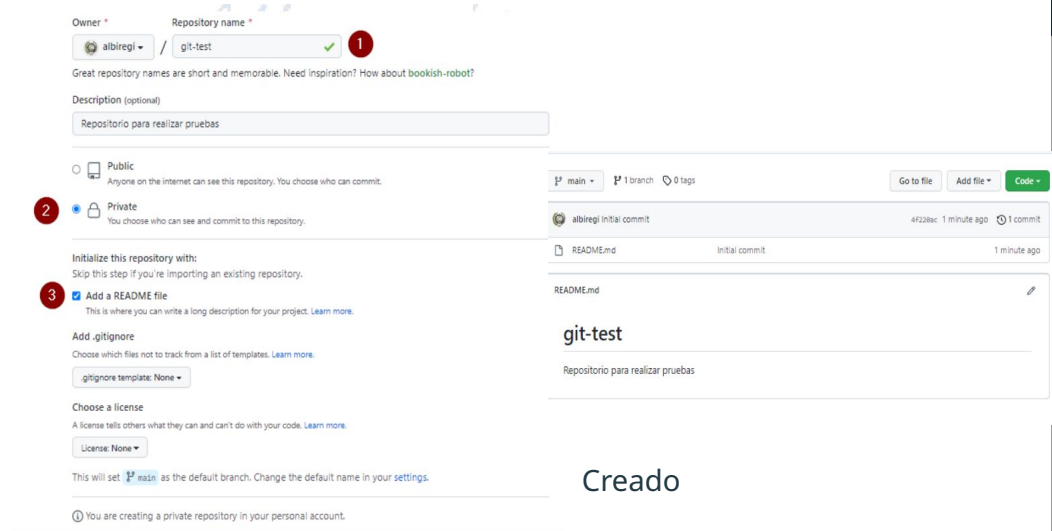
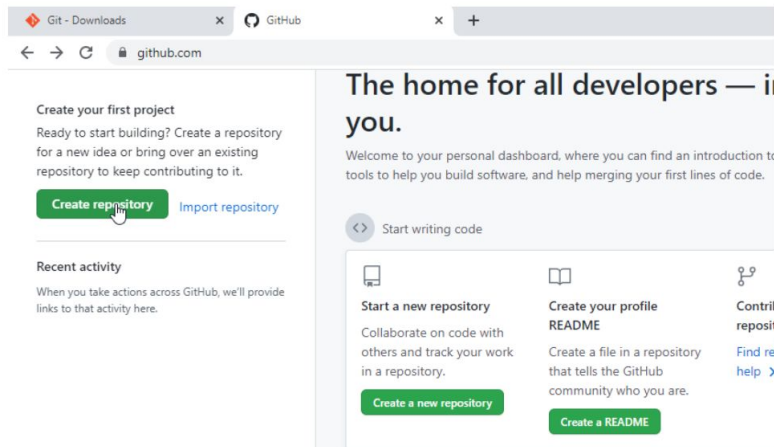
6.1-Comandos básicos de Git

- git init: Crea un repositorio local desde cero.
- git clone: descarga un repositorio existente.
- git add <file>: agrega archivos al área de ensayo.</file>
- git commit m "Descripción": confirma los cambios en el repositorio local.
- git log: muestra los registros de confirmación.
- git status: comprueba el estado de tus archivos.
- git push: Inserta los cambios del repositorio local al repositorio remoto.
- git diff: muestra los cambios en los archivos.
- git checkout <branch>: Cambia entre ramas.</branch>

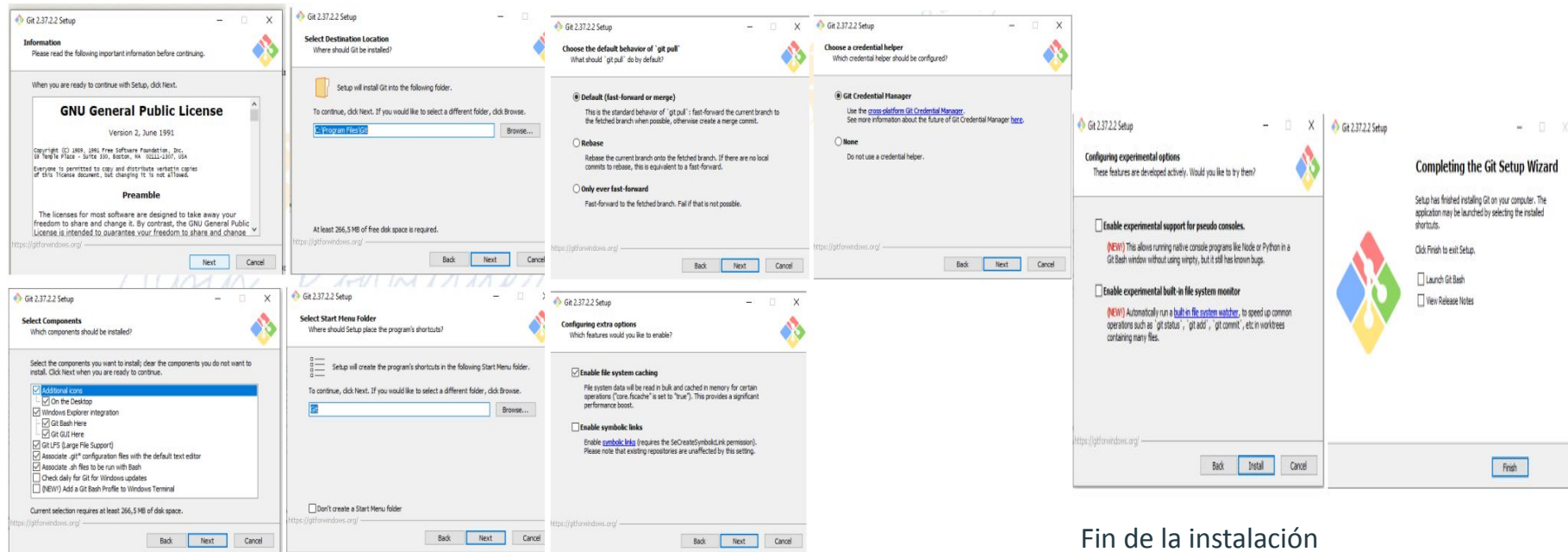


7.-Cómo crear un repositorio en GitHub

Lo primero que vamos a hacer es crear una cuenta de GitHub;



8.-Instalación de Git



Fin de la instalación

9-.Configuración de Git

MINGW64:/d/Git-test

```
regina@DESKTOP-3FF7K39 MINGW64 /d/Git-test
$ |
```

MINGW64:/d/Git-test

```
regina@DESKTOP-3FF7K39 MINGW64 /d/Git-test
$ git version
git version 2.37.2.windows.2
```

git config --global user.name

```
regina@DESKTOP-3FF7K39 MINGW64 /d/Git-test
git config --global user.name "regina"
```

```
regina@DESKTOP-3FF7K39 MINGW64 /d/Git-test
$ git config --global user.name
regina
```

```
regina@DESKTOP-3FF7K39 MINGW64 /d/Git-test
$ git config --global user.email "albiregi@gmail.com"
git config --global user.email "correo_personal"
```

```
regina@DESKTOP-3FF7K39 MINGW64 /d/Git-test
$ git config --list
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager-core
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=master
user.name=regina
user.email=albiregi@gmail.com
```

git config --list

```
regina@DESKTOP-3FF7K39 MINGW64 /d/Git-test
$ ls
'folder 1/'
```

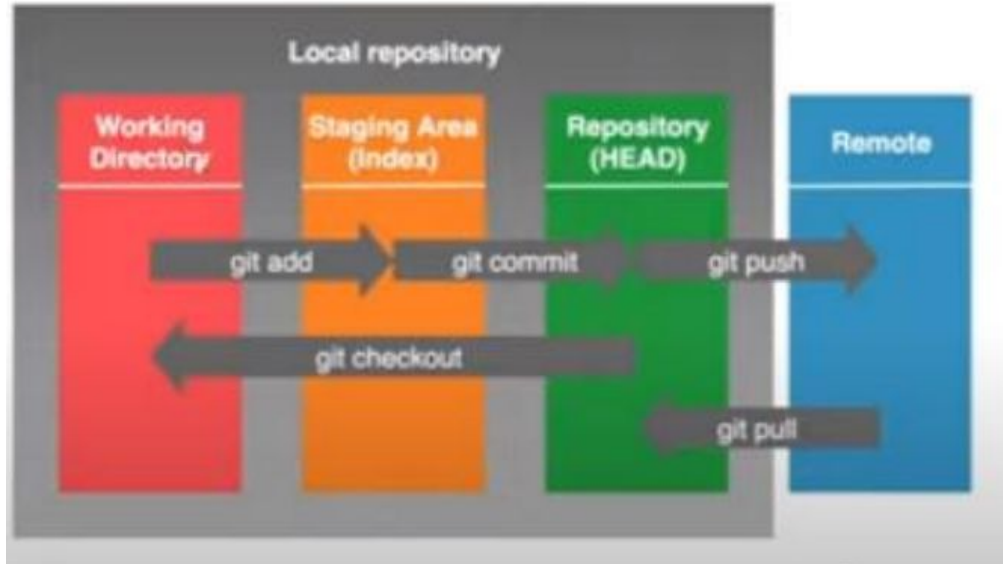
```
regina@DESKTOP-3FF7K39 MINGW64 /d/Git-test
$ ls
total 0
drwxr-xr-x 1 regina 197121 0 Aug 19 14:04 'folder 1/'
```

Para ver con más detalle lo almacenado en nuestra carpeta:ll

```
regina@DESKTOP-3FF7K39 MINGW64 /d/Git-test
$ ls -alh
total 16K
drwxr-xr-x 1 regina 197121 0 Aug 19 14:04 ./
drwxr-xr-x 1 regina 197121 0 Aug 19 12:52 ../
drwxr-xr-x 1 regina 197121 0 Aug 19 14:04 'folder 1/'
```

Para ver los archivos y directorios ocultos:ls -alh

10.-Cómo crear repositorios desde cero y almacenar cambios en nuestro repositorio local



10.1-Git init

```
regina@DESKTOP-3FF7K39 MINGW64 /d/Git-test
$ git init
Initialized empty Git repository in D:/Git-test/.git/

regina@DESKTOP-3FF7K39 MINGW64 /d/Git-test (master)
$
```

Con esto ya tenemos un repositorio hecho “máster”. En nuestra carpeta

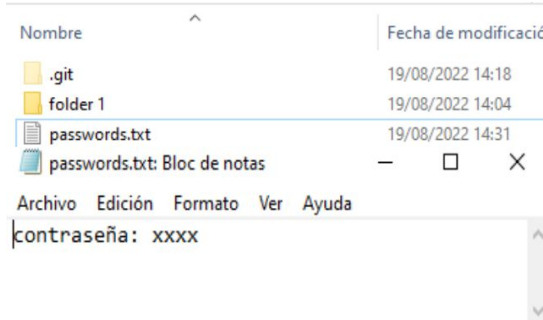
```
regina@DESKTOP-3FF7K39 MINGW64 /d/Git-test (master)
$ ls -alh
total 20K
drwxr-xr-x 1 regina 197121 0 Aug 19 14:18 ./
drwxr-xr-x 1 regina 197121 0 Aug 19 12:52 ../
drwxr-xr-x 1 regina 197121 0 Aug 19 14:18 .git/
drwxr-xr-x 1 regina 197121 0 Aug 19 14:04 'folder 1'/
```

Índice



10.2-.Gitignore

Vamos a crear un archivo llamado passwords



```
regina@DESKTOP-3FF7K39 MINGW64 /d/Git-test (master)
$ vim .gitignore
```

Archivo .gitignore para que cuando indiquemos los cambios a nuestro repositorio remoto, no mande los cambios realizados en este archivo.

```
passwords.txt
.gitignore[+] [unix] (00:59 01/01/1970)
```

Lo no queremos que se muestran en el repositorio remoto cuando realicemos un commit.

```
regina@DESKTOP-3FF7K39 MINGW64 /d/Git-test (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        folder 1/

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

Ya tenemos todo esto listo para ser agregado al Staging Area (Index)

```
regina@DESKTOP-3FF7K39 MINGW64 /d/Git-test (master)
$ git status
On branch master

No commits yet

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

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

Ver el status de mi repositorio local,

10.3-.Git add

```
regina@DESKTOP-3FF7K39 MINGW64 /d/Git-test (master)  
$ git add -A
```

git add -A

```
regina@DESKTOP-3FF7K39 MINGW64 /d/Git-test (master)  
$ git add -A
```

Ver el estado de mi git, solo nos indica que ahora hay que hacer commit.

```
regina@DESKTOP-3FF7K39 MINGW64 /d/Git-test (master)  
$ git status  
On branch master  
  
No commits yet  
  
Changes to be committed:  
  (use "git rm --cached <file>..." to unstage)  
    new file:   .gitignore  
    new file:   folder 1/test.txt
```

Índice



10.4-.Git rm

El archivo test.txt no queremos hacerle un commit, modificarlo antes,

```
regina@DESKTOP-3FF7K39 MINGW64 /d/Git-test (master)
$ git rm --cached 'folder 1'/test.txt
rm 'folder 1/test.txt'
```

```
regina@DESKTOP-3FF7K39 MINGW64 /d/Git-test (master)
$ git status
On branch master

No commits yet

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

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        folder 1/
```

Índice

← Padre

A continuación hacemos un git add. Podemos indicar el archivo en particular al que queremos hacer un add o poner add.

```
regina@DESKTOP-3FF7K39 MINGW64 /d/Git-test (master)
$ git add 'folder 1'/test.txt
warning: in the working copy of 'folder 1/test.txt', LF will be replaced by CRLF
the next time Git touches it
```

```
regina@DESKTOP-3FF7K39 MINGW64 /d/Git-test (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   .gitignore
        new file:   folder 1/test.txt
```

10.5-.Git commit

Ahora vamos a proceder a hacer la transición del Staging Area al Repositorio Local para ello utilizamos el comando `git commit -m "comentario"`.

```
regina@DESKTOP-3FF7K39 MINGW64 /d/Git-test (master)
$ git commit -m "commit inicial"
[master (root-commit) 8f8d4fa] commit inicial
2 files changed, 2 insertions(+)
create mode 100644 .gitignore
create mode 100644 folder 1/test.txt
```

`git commit -m "commit inicial"`

```
regina@DESKTOP-3FF7K39 MINGW64 /d/Git-test (master)
$ git status
On branch master
nothing to commit, working tree clean
```

Observamos que ya todos los archivos y carpetas están en el repositorio local listos para ser enviados al repositorio remoto.

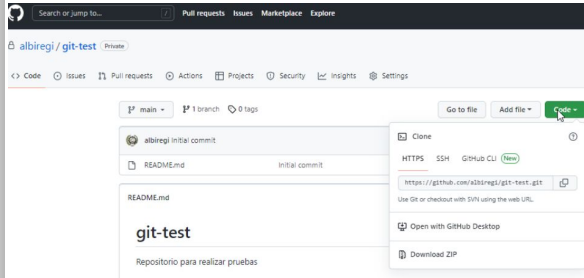
```
regina@DESKTOP-3FF7K39 MINGW64 /d/Git-test (master)
$ git log
commit 8f8d4fa6cf0b846a0e194fdb68f94fb7e101403e (HEAD -> master)
Author: regina <albiregi@gmail.com>
Date:   Fri Aug 19 16:43:12 2022 +0200

    commit inicial
```

Para ver los commit que se han ido realizando utilizaremos el siguiente comando

`git log`

10.6-.Git remote conexión repositorio local con uno remoto



Dirección https con la que nos vamos a comunicar con el repositorio remoto

```
regina@DESKTOP-3FF7K39 MINGW64 /d/Git-test (master)  
$ git remote add origin https://github.com/albiiregi/git-test.git
```

Vamos a enviar nuestros cambios de manera remota.

```
git remote add origin  
https://github.com/albiiregi/git-test.git
```

```
regina@DESKTOP-3FF7K39 MINGW64 /d/Git-test (master)  
$ git branch -M main
```

git branch -M main

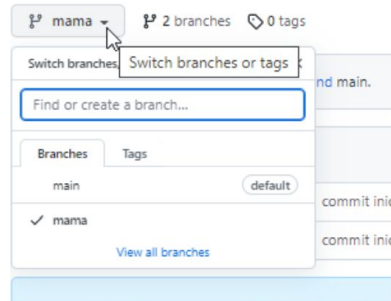
Índice

Padre

```
regina@DESKTOP-3FF7K39 MINGW64 /d/Git-test (main)  
$ git push -u origin main
```

git push -u origin main

```
regina@DESKTOP-3FF7K39 MINGW64 /d/Git-test (main)  
$ git push -u origin main  
Enumerating objects: 6, done.  
Counting objects: 100% (6/6), done.  
Delta compression using up to 8 threads  
Compressing objects: 100% (2/2), done.  
Writing objects: 100% (5/5), 375 bytes | 375.00 KiB/s, done.  
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0  
To https://github.com/albiiregi/git-test.git  
4f220ac..11b9f76 main -> main  
branch 'main' set up to track 'origin/main'.
```



F5 y ya vemos el repositorio remoto

10.7-.Git mv

Para cambiar el nombre de un fichero o directorio de tu repositorio.

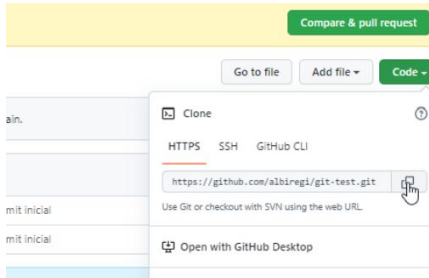
```
git mv test.txt prueba.txt  
git commit -am "He modificado el nombre del fichero test.txt"  
git push
```

Índice

← Padre

10.8-.Git clone

Creamos una carpeta llamada Git y abrimos ahí Git Bash Here.

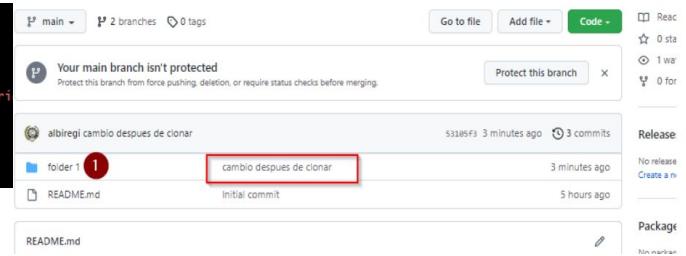


```
regina@DESKTOP-3FF7K39 MINGW64 /d/git
$ cd git-test

regina@DESKTOP-3FF7K39 MINGW64 /d/git/git-test (main)
$ git log
commit 4f220ac31c0aabe084d3e4db716c8547b36eebae (HEAD -> main, origin/main, ori
in/HEAD)
Author: albiiregi <111572134+albiiregi@users.noreply.github.com>
Date:   Fri Aug 19 12:59:10 2022 +0200

    Initial commit
```

Modificar el contenido del fichero
que hay dentro de folder 1



```
regina@DESKTOP-3FF7K39 MINGW64 /d/git
$ git clone https://github.com/albiiregi/git-test.git
Cloning into 'git-test'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 8 (delta 0), reused 5 (delta 0), pack-reused 0
Receiving objects: 100% (8/8), done.
```

```
regina@DESKTOP-3FF7K39 MINGW64 /d/git/git-test (main)
$ git add -A

regina@DESKTOP-3FF7K39 MINGW64 /d/git/git-test (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   folder 1/.test.txt.swp
    modified:   folder 1/test.txt
    deleted:    passwords.txt
```

Observamos que ya está
actualizado el repositorio remoto.

Veremos que está alojado el repositorio
que acabamos de clonar.

Índice

Hacemos el commit



10.9-.Repositorio remoto compartido

Supongamos que en un repositorio están trabajando varias personas y una de ellas realiza un commit y no tenemos ese cambio almacenado en nuestro ordenador. ¿Qué debemos hacer? Guardar este cambio con el comando git pull.

```
regina@DESKTOP-3FF7K39 MINGW64 /d/git/git-test (main)
$ git pull origin main
From https://github.com/albiregi/git-test
* branch      main      -> FETCH_HEAD
Already up to date.
```

Índice

← Padre

10.10-.Git Branch

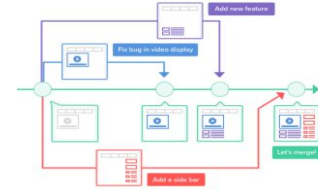
En un entorno colaborativo, es habitual que varios desarrolladores compartan y trabajen sobre el mismo código fuente. Mientras que algunos desarrolladores corregirán errores, otros implementarán nuevas funciones, etc. Con tantas cosas sucediendo, es necesario que exista un sistema para administrar diferentes versiones de la misma base de código.

Índice

← Padre

10.10.1-¿Qué es una rama de Git?

Una rama de git es una línea de desarrollo independiente tomada del mismo código fuente.



```
regina@DESKTOP-3FF7K39 MINGW64 /d/git/git-test (main)
$ git branch nuevo-feature
```

```
git branch nuevo-feature
```

```
regina@DESKTOP-3F7K39 MINGW64 /d/git/git-test (main)
$ git checkout nuevo-feature
Switched to branch 'nuevo-feature'

regina@DESKTOP-3F7K39 MINGW64 /d/git/git-test (nuevo-feature)
git checkout nuevo-feature
```

```
Regina@DESKTOP-3FF7K39 MINGW64 /d/git/git-test (nuevo-feature)
$ git branch
main
* nuevo-feature
```

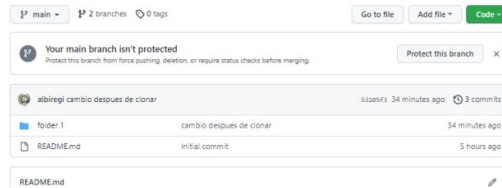
git branch



Índice

```
regina@DESKTOP-3FF7K39 MINGW64 /d/git/git-test (nuevo-Feature)
$ git branch -a
main
* nuevo-Feature
remotes/origin/HEAD -> origin/main
remotes/origin/main
```

```
git branch -a
```



Veremos que la rama nuevo-feature no está aún, ya que no hemos hecho ningún push.

```
MINGW64:/d/git/git-test
Hacemos cambios
Segundo cambio despues de clonar
Cambio en branch nuevo-feature
[...]
```

Realizamos una modificación en el archivo test.txt del directorio 'folder 1'.

```
regina@DESKTOP-3FF7K39 MINGW64 /d/git/git-test (nuevo-feature)
$ git add -A

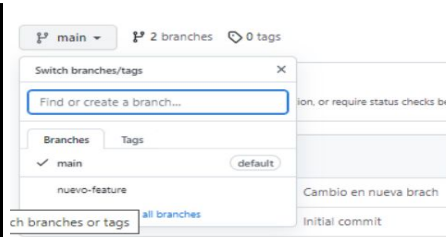
regina@DESKTOP-3FF7K39 MINGW64 /d/git/git-test (nuevo-feature)
$ git commit -m "Cambio en nueva brach"
[nuevo-feature ea48fc3] Cambio en nueva brach
1 file changed, 2 insertions(+), 1 deletion(-)
```

```
regina@DESKTOP-3FF7K39 MINGW64 /d/git/git-test (nuevo-feature)
$ git status
On branch nuevo-feature
nothing to commit, working tree clean
```


10.11-.Git push

Ya solo nos queda reflejar estos cambios en el repositorio remoto, pero en la rama nuevo-feature.

```
regina@DESKTOP-3FF7K39 MINGW64 /d/git/git-test (nuevo-feature)
$ git push origin nuevo-feature
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 426 bytes | 426.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'nuevo-feature' on GitHub by visiting:
remote:   https://github.com/albiregi/git-test/pull/new/nuevo-feature
remote:
To https://github.com/albiregi/git-test.git
 * [new branch]    nuevo-feature -> nuevo-feature
```



```
regina@DESKTOP-3FF7K39 MINGW64 /d/git/git-test (nuevo-feature)
$ git branch -a
main
* nuevo-feature
remotes/origin/HEAD -> origin/main
remotes/origin/main
remotes/origin/mama
remotes/origin/nuevo-feature
```

`git push origin nuevo-feature`

Observad que el contenido del archivo test.txt de la rama nuevo-feature es distinto al de la rama main.

Podemos observar que la rama nuevo-feature, ya está en el repositorio remoto.

`git branch -a`

Índice

Padre

10.12-.Git pull

Lo último es combinar los cambios de la rama nuevo-feature para que todos los cambios queden en la línea principal, main. Lo primero que debemos realizar es cambiarnos a la rama principal, main.

```
regina@DESKTOP-3FF7K39 MINGW64 /d/git/git-test (nuevo-feature)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

regina@DESKTOP-3FF7K39 MINGW64 /d/git/git-test (main)
```

git checkout main

```
regina@DESKTOP-3FF7K39 MINGW64 /d/git/git-test (main)
$ git pull origin main
From https://github.com/albiregi/git-test
 * branch      main      -> FETCH_HEAD
Already up to date.
```

Asegurarnos que no ha habido ningún cambio en la rama main

git pull origin main

Índice

← Padre

10.13-.Git merge

Vamos utilizar el comando merge para validar las brach con las que se ha hecho un merge.

```
regina@DESKTOP-3FF7K39 MINGW64 /d/git/git-test (main)
$ git branch --merged
* main
```

`git branch --merged`

Hacemos el merge en main de la rama nuevo-feature.

```
regina@DESKTOP-3FF7K39 MINGW64 /d/git/git-test (main)
$ git merge nuevo-feature
Updating 53105f3..ea48fc3
Fast-forward
 folder 1/test.txt | 3 ++-
 1 file changed, 2 insertions(+), 1 deletion(-)
```

`git merge nuevo-feature`

```
regina@DESKTOP-3FF7K39 MINGW64 /d/git/git-test (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```

Los cambios están en local, pero no en remoto.
Último paso, hacer un push para
reflejar los cambios locales en remoto

```
regina@DESKTOP-3FF7K39 MINGW64 /d/git/git-test (main)
$ git push origin main
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/albiregi/git-test.git
 53105f3..ea48fc3  main -> main
```

`git push origin main`

Índice

← Padre

main	git-test / folder 1 / test.txt
albiregi	Cambio en nueva brach
Latest	
At 1 contributor	
3 lines (3 sloc) 88 Bytes	
1	Hacemos cambios
2	Segundo cambio despues de clonar
3	Cambio en branch nuevo-feature

10.14-.Eliminar un branch

Una vez que se ha realizado el merge, lo aconsejable es eliminar la rama para evitar sobrecargar nuestro repositorio.

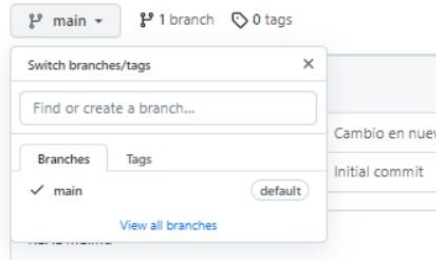
```
regina@DESKTOP-3FF7K39 MINGW64 /d/git/git-test
$ git branch -a
* main
nuevo-feature
remotes/origin/HEAD -> origin/main
remotes/origin/main
remotes/origin/nuevo-Feature
```

```
regina@DESKTOP-3FF7K39 MINGW64 /d/git/git-test (main)
$ git branch -a
* main
nuevo-feature
remotes/origin/HEAD -> origin/main
remotes/origin/main
```

Vamos a borrar la rama tanto local como remota nuevo-feature

`git push origin -delete nuevo-feature`

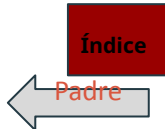
Nos queda eliminar la rama a nivel local.



```
regina@DESKTOP-3FF7K39 MINGW64 /d/git/git-test (main)
$ git branch -d nuevo-feature
Deleted branch nuevo-feature (was ea48fc3).

regina@DESKTOP-3FF7K39 MINGW64 /d/git/git-test (main)
$ git branch -a
* main
remotes/origin/HEAD -> origin/main
remotes/origin/main
```

`git branch -d nuevo-feature`



Gracias por su tiempo.Espero que le haya gustado la presentación:)

Fin

