1.- Configuración inicial

Configuración necesaria para cada commit que hacemos:

```
1. git config --global user.name "Your Name"
2. git config --global user.email "youremail@domain.com"
```

2.- Uso de GitHub

3. Nos registramos en Github.

Configuración para establecer la comunicación entre un repositorio local y uno remoto mediante ssh

De esta forma evitaremos introducir usuario/contraseña en nuestra consola.

4. Accedemos a nuestra cuenta de GitHub.
5. Vamos a los *settings* del **USUARIO** y asociamos una ssh-key.
6. Como creamos nuestra ssh-key:

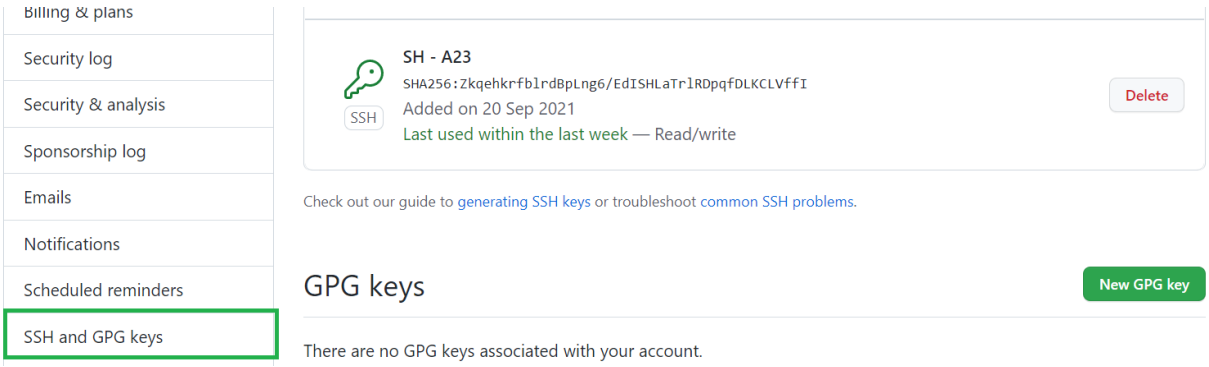
```
ssh-keygen
```

7. Copiaremos el contenido de ~/.ssh/id_rsa.pub a una nueva clave ssh en GitHub

```
alumno@dws:~$ mkdir .ssh
alumno@dws:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/alumno/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/alumno/.ssh/id_rsa
Your public key has been saved in /home/alumno/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:eDbjwS5mk85AKNbM7CpDO9efqVsmiuFFDZ8DsGY+GB8 alumno@dws
The key's randomart image is:
```

```
alumno@dws:~$ cd .ssh
alumno@dws:~/.ssh$ ls
id_rsa  id_rsa.pub
```

```
alumno@dws:~/.ssh$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDeSpYrokkC9QfKfbCH0ONQ7OY3UCzUAxZW/lTUlghB
KAZuXSVLD23KZ2wNajSFIOZzm9tBMNv6OhKwt61fKcdGP6vMmu5NOdnF+2Gblw6dW7wX4HmW/OISuWoC
PUEP7ZTzr9gsWdRbe+7tvTCEKcMy8dZBlPh807u+1qIgtZ2wcnhp/9VNgoGZF/RXRPMtO+2ep+6EZlkP
Fn1TXALSPAKHxWtMQIsSFdqIhm9XwPcEDhtIjd8uYyOAHWi9WZuiGjJMmDWdnLhRchy355h3ujbJf0lO
```



3.- Trabajar con el repositorio localmente

El comando **`git init`** permite iniciar el repositorio. Nos situamos en nuestro directorio de trabajo y tecleamos lo siguiente:

8. `git init`
9. `echo "# Repositorio de ejemplo" >> README.md`
10. `git add README.md`
11. `git commit -m "Commit inicial"`

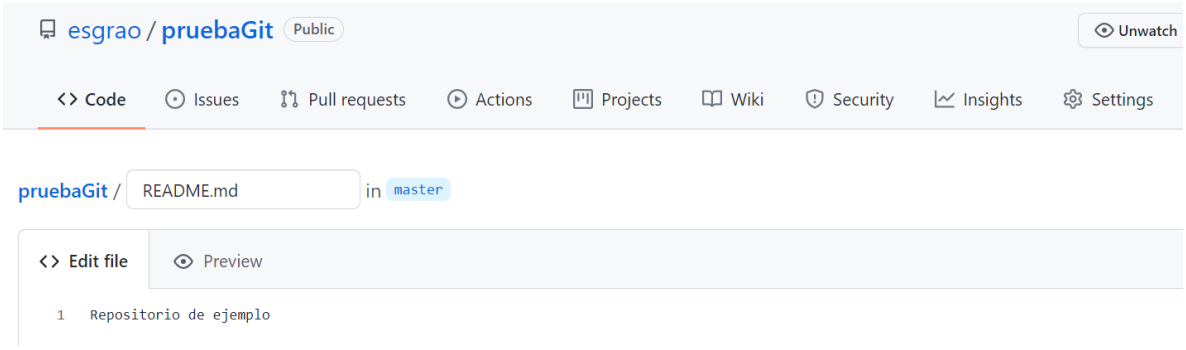
```
alumno@dws:~/dws/pruebaGit$ git init
Inicializado repositorio Git vacío en /home/alumno/dws/pruebaGit/.git/
alumno@dws:~/dws/pruebaGit$ echo "Commit inicial" >> README.md
alumno@dws:~/dws/pruebaGit$ git add README.md
alumno@dws:~/dws/pruebaGit$ git commit -m 'Commit incial'
[master (commit-raíz) c1d9ec5] Commit incial
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
```

4.- Trabajar con un repositorio remoto

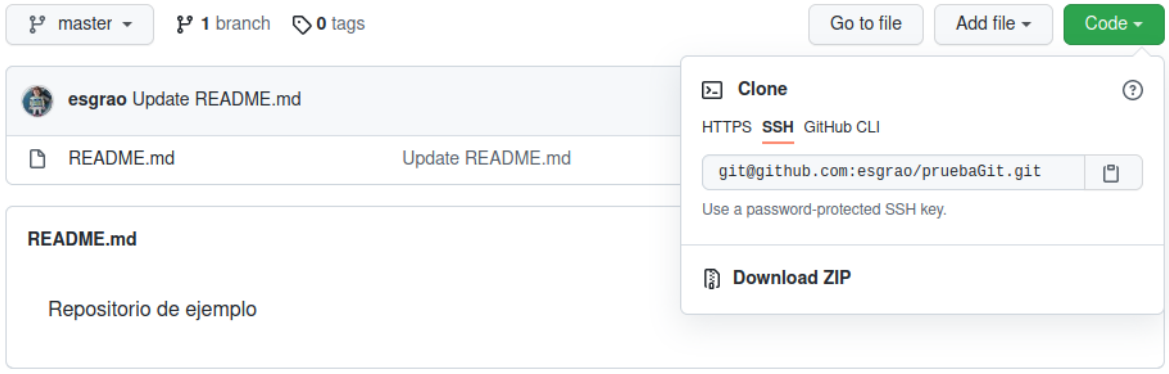
También es posible partir de un repositorio nuestro o ajeno ya creado utilizando el comando git clone.

- `git clone <url repo>  #habitual`
- `git clone <url repo> <carpeta> #permite definir destino`

12. Crea un nuevo repositorio en GitHub llamado *pruebaGit*.
13. Añade un nuevo fichero llamado README.md con el siguiente contenido:



14. Clona el repositorio remoto en local utilizando git clone.




```
alumno@dws:~$ git clone git@github.com:esgrao/pruebaGit.git
Clonando en 'pruebaGit'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
Recibiendo objetos: 100% (6/6), listo.
alumno@dws:~$
```

Ver el estado

Para consultar el estado del repositorio el comando `git status` nos dirá qué ficheros hay nuevos o modificados. La rama en la que estamos, por defecto *master*. Y si estamos por detrás/delante del origin, el repositorio remoto.

15. Accedemos al repositorio clonado:

```
alumno@dws:~$ cd pruebaGit
alumno@dws:~/pruebaGit$ ls
README.md
```

16. Creamos un nuevo fichero llamado casa.md.
17. Consultamos su estado:

```
alumno@dws:~/pruebaGit$ git status
En la rama master
Tu rama está actualizada con 'origin/master'.

Archivos sin seguimiento:
  (usa "git add <archivo>..." para incluirlo a lo que se será confirmado)
        casa.md

no hay nada agregado al commit pero hay archivos sin seguimiento presentes (us
a "git add" para hacerles seguimiento)
```

Preparar y comprometer cambios

Con **git add** pasamos los ficheros modificados/nuevos a **preparados**. Tras hacer esto se guardan temporalmente los cambios realizados.

```
git add <nombre fichero>     #un sólo fichero
git add img/logo.jpg
git add <nombre directorio> #todo lo de un directorio
git add .                    #caso particular, todo, todo...
```

Con `**git commit** -m "...." comprometemos los cambios en nuestro repositorio local.

```
git add .
git commit -m "comentario explicativo"
```

18. Modifica el fichero casa.md. y comprueba su estado:

```
alumno@dws:~/pruebaGit$ echo "-mas cambios" >> casa.md
alumno@dws:~/pruebaGit$ git status
En la rama master
Tu rama está actualizada con 'origin/master'.

Archivos sin seguimiento:
  (usa "git add <archivo>..." para incluirlo a lo que se será confirmado)
        casa.md

no hay nada agregado al commit pero hay archivos sin seguimiento presentes (us
a "git add" para hacerles seguimiento)
```

19. Pasa el nuevo fichero a estado preparado y comprueba su estado:

```
alumno@dws:~/pruebaGit$ git add casa.md
alumno@dws:~/pruebaGit$ git status
En la rama master
Tu rama está actualizada con 'origin/master'.

Cambios a ser confirmados:
  (usa "git restore --staged <archivo>..." para sacar del área de stage)
        nuevo archivo:  casa.md
```

20. Añade una línea nueva al fichero casa.md y comprueba su estado:

```
alumno@dws:~/pruebaGit$ echo "-mas cambios" >> casa.md
alumno@dws:~/pruebaGit$ git status
En la rama master
Tu rama está actualizada con 'origin/master'.

Cambios a ser confirmados:
  (usa "git restore --staged <archivo>..." para sacar del área de stage)
        nuevo archivo:  casa.md

Cambios no rastreados para el commit:
  (usa "git add <archivo>..." para actualizar lo que será confirmado)
  (usa "git restore <archivo>..." para descartar los cambios en el directorio
de trabajo)
        modificado:     casa.md
```

21. Compromete los cambios en el repositorio local:

```
alumno@dws:~/pruebaGit$ git commit -m 'Guardar cambios casa'
[master e064145] Guardar cambios casa
 1 file changed, 2 insertions(+)
 create mode 100644 casa.md
alumno@dws:~/pruebaGit$ git status
En la rama master
Tu rama está adelantada a 'origin/master' por 1 commit.
  (usa "git push" para publicar tus commits locales)

Cambios no rastreados para el commit:
  (usa "git add <archivo>..." para actualizar lo que será confirmado)
  (usa "git restore <archivo>..." para descartar los cambios en el directorio
de trabajo)
        modificado:     casa.md

sin cambios agregados al commit (usa "git add" y/o "git commit -a")
```

Marcha atrás de un fichero modificado

Hemos modificado un fichero y queremos recuperar la anterior versión:

```
git checkout <nombrefichero>
git checkout casa.md
git checkout <nombre directorio>
git checkout .
```

22. Añade una línea al fichero casa.md y descarta los cambios utilizando checkout:

```
alumno@dws:~/pruebaGit$ echo "-mas cambios v3" >> casa.md
alumno@dws:~/pruebaGit$ git checkout casa.md
Actualizada 1 ruta desde el index
alumno@dws:~/pruebaGit$ cat casa.md
```

Marcha atrás de un fichero preparado

Queremos sacar del estado preparado después de **git add**:

```
git reset HEAD <nombrefichero>
git reset HEAD casa.md
git reset HEAD <nombre directorio>
git reset HEAD .
```

23. Revierte los cambios de un fichero preparado:

```
alumno@dws:~/pruebaGit$ echo "-mas cambios v4" >> casa.md
alumno@dws:~/pruebaGit$ git add casa.md
alumno@dws:~/pruebaGit$ git reset HEAD casa.md
Cambios fuera del área de stage tras el reset:
M       casa.md
```

5.- Guardar en remoto

Git push

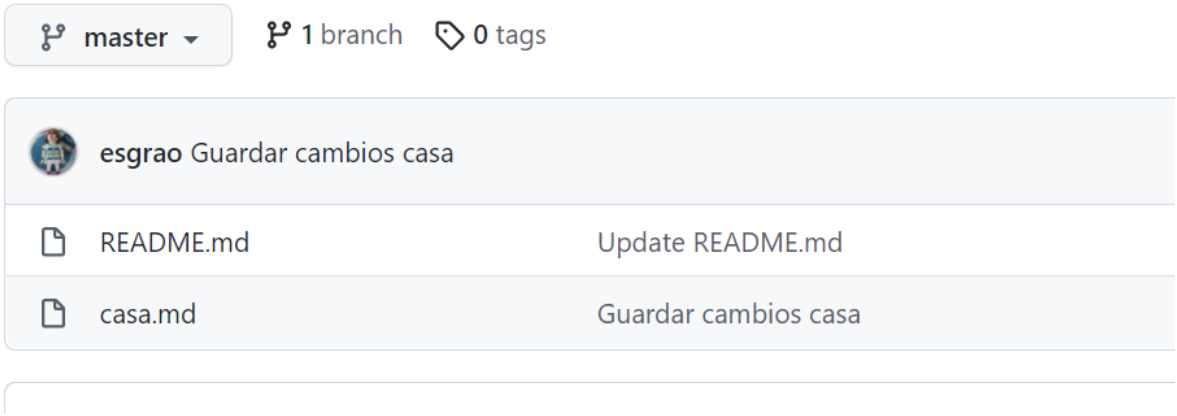
Push es el comando usado para subir código a *origin*, es decir el repositorio **remoto**.

- ***Origin*** es el nombre usado por defecto para la copia remota.
- ***Master*** es el nombre de la rama por defecto.
- Ejemplos:

```
git push <repo remoto> <rama>    # sube una rama concreta
git push origin dev              # Ej. sube rama dev
git push -u <repo remoto> <rama># sube y predetermina rama
git push -u origin master        # Ej. sube y pred. Master
```

24. Sube los cambios del repositorio local al remoto y comprueba que existe el fichero casa.md en GitHub:

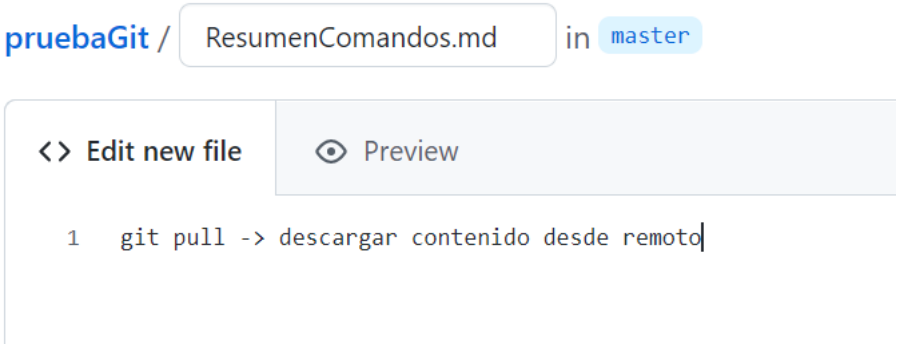
```
alumno@dws:~/pruebaGit$ git push
Enumerando objetos: 4, listo.
Contando objetos: 100% (4/4), listo.
Compresión delta usando hasta 4 hilos
Comprimiendo objetos: 100% (2/2), listo.
Escribiendo objetos: 100% (3/3), 307 bytes | 307.00 KiB/s, listo.
Total 3 (delta 0), reusado 0 (delta 0)
To github.com:esgrao/pruebaGit.git
   3cb8a11..e064145  master -> master
```



Git pull

Extraer y descargar contenido desde un repositorio.

25. Añade un fichero en el repositorio remoto:



26. Actualiza el repositorio local utilizando git pull:

```
alumno@dws:~/pruebaGit$ git pull
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
Desempaquetando objetos: 100% (3/3), 736 bytes | 147.00 KiB/s, listo.
Desde github.com:esgrao/pruebaGit
   e064145..a8a916d  master     -> origin/master
Actualizando e064145..a8a916d
Fast-forward
 ResumenComandos.md | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 ResumenComandos.md
alumno@dws:~/pruebaGit$ ls
casa.md  README.md  ResumenComandos.md
```

27. Sal del repositorio actual.

Fork & Pull Request

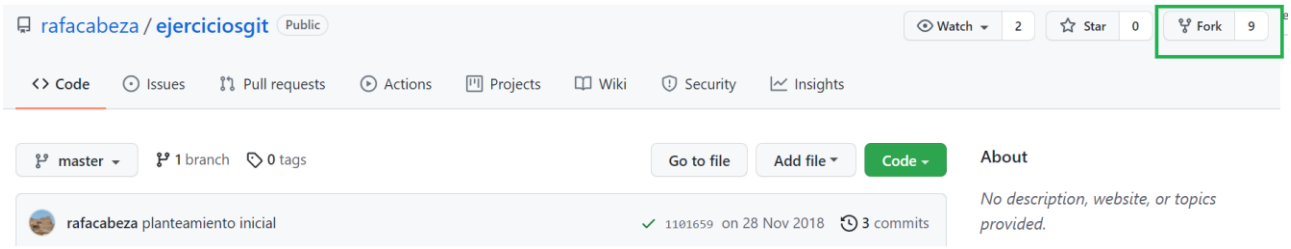
Fork crea un clon de un repositorio ajeno en nuestro espacio.

- Es una funcionalidad de GitHub/Bitbucket.
- El clon es de mi propiedad y lo puedo clonar y editar.

Pull Request es una solicitud para que el dueño original incorpore mis cambios.

- Se hace en la interfaz web.
- El dueño original lo acepta o no...

28. Desde la interfaz gráfica de GitHub haz un fork de este repositorio: rafacabeza/ejerciciosgit.



29. Edita los settings de tu repositorio y habilita "GitHub Pages"
30. Comprueba que funcionan: https://esgrao.github.io/ejerciciosgit/
    a. Debes cambiar la url con tu nombre de usuario.

6.- Ramas (branch)

Por defecto trabajamos en la rama master. Las ramas se usan fundamentalmente para separar tareas sin modificar la rama master. Una vez terminada la tarea se funde (**merge**) la rama de tarea con la rama master. Esto permite cambiar de rama/tarea y dejarla incompleta sin dejar el proyecto en versiones incompletas o inestables.

- Comandos con ramas:

```
git branch              // lista de ramas
git branch  <rama>      // crear rama
git checkout <rama>     // cambiar de rama
git checkout -b <rama>  // crear y cambiar rama 2 en 1
git branch -d <rama>    // borrar rama
```

- Fundir ramas:

```
git checkout master     // nos ponemos en rama master
git merge <rama>        //fundimos con la rama deseada
```

31. Clona el repositorio ejerciciosgit y crea una rama:

```
alumno@dws:~$ git clone git@github.com:esgrao/ejerciciosgit.git
Clonando en 'ejerciciosgit'...
remote: Enumerating objects: 11, done.
remote: Total 11 (delta 0), reused 0 (delta 0), pack-reused 11
Recibiendo objetos: 100% (11/11), listo.
```

```
alumno@dws:~$ git clone git@github.com:esgrao/ejerciciosgit.git
Clonando en 'ejerciciosgit'...
remote: Enumerating objects: 11, done.
remote: Total 11 (delta 0), reused 0 (delta 0), pack-reused 11
Recibiendo objetos: 100% (11/11), listo.
alumno@dws:~$ git branch tarea1
fatal: no es un repositorio git (ni ninguno de los directorios superiores):
alumno@dws:~$ cd ejerciciosgit
alumno@dws:~/ejerciciosgit$ git branch tarea1
alumno@dws:~/ejerciciosgit$ git branch
* master
  tarea1
```

32. Sitúate en la nueva rama y añade un fichero:

```
alumno@dws:~/ejerciciosgit$ git checkout tarea1
Cambiado a rama 'tarea1'
alumno@dws:~/ejerciciosgit$ git branch
  master
* tarea1
```

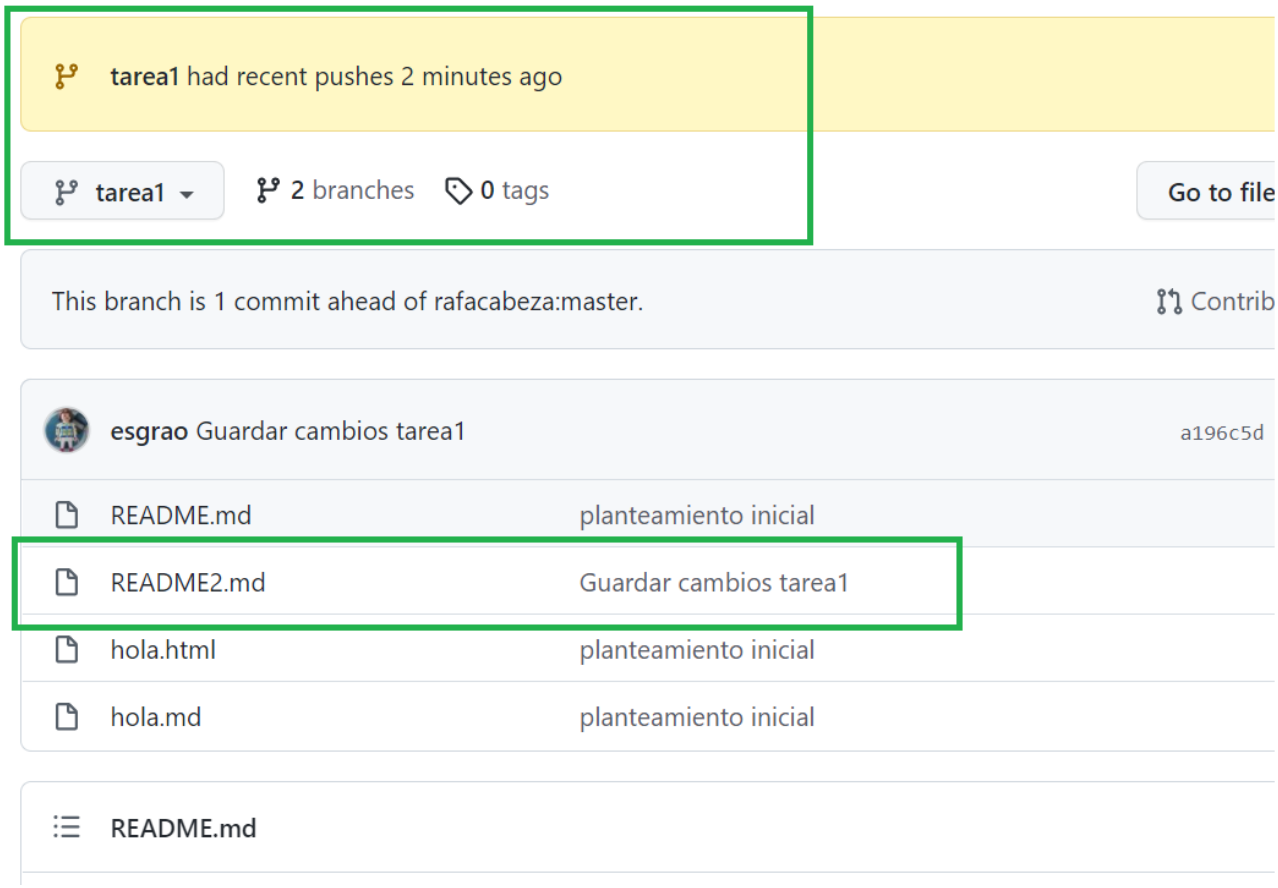
```
alumno@dws:~/ejerciciosgit$ echo "Tarea 1 crear una rama" >> README2.md
alumno@dws:~/ejerciciosgit$ ls
hola.html  hola.md  README2.md  README.md
```

33. Sube los cambios al repositorio local:

```
alumno@dws:~/ejerciciosgit$ git add .
alumno@dws:~/ejerciciosgit$ git commit -m 'Guardar cambios tarea1'
[tarea1 d4139e0] Guardar cambios tarea1
 1 file changed, 1 insertion(+)
```

34. Sube los cambios al repositorio remoto:

```
alumno@dws:~/ejerciciosgit$ git push --set-upstream origin tarea1
Enumerando objetos: 4, listo.
Contando objetos: 100% (4/4), listo.
Compresión delta usando hasta 4 hilos
Comprimiendo objetos: 100% (2/2), listo.
Escribiendo objetos: 100% (3/3), 354 bytes | 354.00 KiB/s, listo.
Total 3 (delta 0), reusado 0 (delta 0)
remote:
remote: Create a pull request for 'tarea1' on GitHub by visiting:
remote:      https://github.com/esgrao/ejerciciosgit/pull/new/tarea1
remote:
To github.com:esgrao/ejerciciosgit.git
 * [new branch]      tarea1 -> tarea1
Rama 'tarea1' configurada para hacer seguimiento a la rama remota 'tarea1'
'origin'.
```



Ramas remotas

Se usan cuando queremos compartir una rama de trabajo.

- Subir una rama:
  - `git push origin <rama>`
- Bajar la rama tiene dos partes:
  - `git fetch  #crea la rama oculta origin/<rama>`
  - `git pull`
- Crear una rama local asociada a origin:
  - `git checkout -b nombreRama`

35. Realiza un clon a tu ordenador y descarga.
36. Modifica la rama de trabajo y sube los cambios.
37. Funde el trabajo en la rama master y súbela.
38. Borra la rama de trabajo en local y en remoto.

7.- Revisando código

Los comandos básicos son log y diff:

git log nos muestra información histórica y soporta una gran cantidad de parámetros:

```
git log             #uso base
git log -<n>        #log de los últimos n commits
git log --oneline -5 #lista de commits breve
git log --follow [file] #lista de commit con cambios para "file"
git show <commit>  # información completa de un commit concreto
```

Git diff sirve para ver las diferencias entre el estado actual y otro:

```
git diff     # diferencia entre estado actual y el preparado o comprometido
git diff --cached #diferencia entre el preparado y el último commit
git diff --stat #idem al primero pero con información resumida de cambios
```

diff desde un commit concreto

Podemos hacer diff sobre todo el repositorio o sobre un fichero.

Podemos referirnos a un commit por su hash o contando hacia atrás desde el HEAD o desde una rama.

```
git diff <commit> # cambios desde un commit hasta la actualidad
git diff <commit> <file># idem fichero concreto
git diff master~2 README.md #Cambios en los dos últimos commit de master
git diff HEAD~2 README.md #Cambios en los dos últimos commit HEAD
git diff cd598e4 README.md #Cambios desde un commit concreto por hash
```

diff entre dos commit

```
git diff <commit>:<file>   <commit2>:<file> #diff de un fichero entre dos commits
git diff cd598e4:README.md  31422ac:README.md
git diff master~20:README.md  master~1:README.md
```

GitIgnorando cosas: .gitignore

Podemos decir a git que no tenga en consideración algunos ficheros/directorios

- Para hacerlo debemos crear un fichero .gitignore en el directorio raíz.