

Guía Rápida

Por: Antonio Martínez

<https://github.com/metantonio>

Última actualización: 26-02-2022

Índice

Convención en la notación.....	3
GitHub	4
0 Instalación de Git Bash	4
1 Crear repositorio	4
2 Agregar archivos para empaquetar y prepararlos para subirlos a GitHub	5
3 Creación de un commit (cómo punto de guardado).....	5
4 Vinculación del repositorio local con el remoto	6
5 Primer Push en el repositorio.....	7
6 Clonar repositorio ya existente en GitHub.....	7
7 Desvincular del remoto, lo repositorios clonados	7
8 Ver Branch (todas ramas creadas)	8
9 Actualizar nuestro repositorio local (actualiza la rama local).....	8
10 Verificamos repositorios vinculados:	9
11 Operaciones comunes con ramas o Branchs:	9
11.1 Crear una nueva rama sin ubicarnos en ella	9
11.2 Ubicarse en una rama para trabajar en ella	9
11.3 Crear una nueva rama y ubicarse en ella automáticamente (sería la combinación de los pasos 11.1 y 11.2).....	9
11.4 Crear una nueva rama y hacerle push directamente con modificaciones que tengamos	10
11.5 Eliminación de una rama (eliminación en el repositorio local)	10
11.6 Eliminación de una rama en el repositorio remoto (GitHub).....	10
12 Reset del repositorio local para que sea igual a una rama remota en GitHub (ejemplo con la rama main. Nota: peligro de perder los cambios locales)	11

13 Merge entre ramas.....	11
14 Flujo de trabajo cooperativo en repositorios.	12

Convención en la notación



lowercase, kebab-case

```
<div class="someclass"></div>  
<div class="some-class"></div>
```



```
<DIV class="somClass"></DIV>  
<Div Class="some_class"></Div>
```



kebab-case, BEM

```
.some-class { ... }  
.some-class__body { ... }  
.some-class—big { ... }  
.some-class__body—big { ... }
```

```
.someClass { ... }  
.SomeClass-body { ... }  
.some-class_BIG { ... }  
.SOME-CLASS__body—big { ... }
```



camelCase, PascalCase, UPPERCASE*

```
let myVariable;  
const MY_CONSTANT_VAR = 2;  
Class MyClass { ... }  
function myFunction( ... ) { ... };
```

```
let my_variable;  
const MY-CONSTANT-VAR= 2;  
Class myClass { ... }  
function my_function( ... ) { ... };
```



snake_case, PascalCase, UPPERCASE*

```
my_variable = []  
CONSTANT_VAR = 2  
class MyClass( ... ):  
def my_function( ... ):
```

```
myVariable = []  
CONSTANT-VAR = 2  
class my_class( ... ):  
def MyFunction( ... ):
```



kebab-case

repo-for-my-project

Repo-for-my-Project

REPO_FOR_MY_PROJECT

repoForMyProject

repo_for_my_project



kebab-case

my-projects-url/user-name

My-Projects-Url/User-name

MY_PROJECT_URL/USER_NAME

myProjectsUrl/userName

my_projects_url/user_name

GitHub

GitHub es una forja para alojar proyectos utilizando el sistema de control de versiones Git. Se utiliza principalmente para la creación de código fuente de programas de ordenador.

0 Instalación de Git Bash

Esto habilitará el comando **git** en powershell, cmd, terminal de Visual Studio Code, etc.

<https://gitforwindows.org/>

1 Crear repositorio

Al crear un repositorio nuevo, el nombre del mismo será el nombre del proyecto y a su vez será parte de la URL del mismo, por lo tanto, se recomienda colocar el repositorio en kebab-case:

[ejemplo-nombre-del-proyecto-en-kebab-case](#)

Los repositorios suelen ser **públicos** de manera predeterminada, pero se puede cambiar a privado; y al crearlos es mejor no inicializar con readme o licencias, de manera que la subida de archivos al repositorio sea limpia.

Al crear el repositorio, se establece como rama principal **main** de manera predeterminada.

Después de creado el repositorio en GitHub, vamos a nuestros archivos en local, y estando dentro de la carpeta raíz de nuestro proyecto, si nunca ha estado en github, debemos inicializar el repositorio de manera local con el siguiente comando. (Asegurarse que la terminal está ubicada en el directorio raíz).

```
git init
```

Luego, de forma local crearemos la rama main:

```
git branch -M main
```

2 Agregar archivos para empaquetar y prepararlos para subirlos a GitHub

Se usa el commando:

```
git add nombre-archivo.extensión
```

Si se quiere agregar todos los archivos simultáneamente (**se recomienda fuertemente esta opción**), se usa:

```
git add .
```

3 Creación de un commit (cómo punto de guardado)

El comando usado será:

git commit -m "aquí va un comentario obligatorio sobre el commit"

Al crear un commit por primera vez en bash, pedirá quién lo está haciendo y el correo de la persona (sólo la primera vez, por lo tanto, habrá que colocarlo). En versiones más nuevas, perderá el username de tu GitHub y el Token.

Al verificar el estado del repositorio, se verá entonces que nuestro repositorio local está a un commit por encima del repositorio remoto recién creado.

git status

```
$ 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
```

Vale destacar que **git status**, puede ser útil para identificar errores en momentos que haya problemas con los commit.

4 Vinculación del repositorio local con el remoto

Ahora vinculamos el repositorio remoto con el local, a través de la **URL** del repositorio recién creado. Usando el comando:

git remote add origin https://github.com/usuario/nombre-del-repositorio

Verificamos que el repositorio local esté vinculado con el remoto, tanto para hacer fetch, como para hacer push; con el comando:

```
git remote -v
```

5 Primer Push en el repositorio

Únicamente durante el **primer push**, se usará el siguiente comando:

```
git push -u origin main
```

Cualquier siguiente subida que hagamos en la rama **main**, será suficiente con:

```
git push
```

6 Clonar repositorio ya existente en GitHub

En la terminal apuntamos al directorio dónde queremos bajar el repositorio, escribiendo:

```
git clone https:.....github.com/...../nombre-proyecto
```

Se debe ingresar a la carpeta local del repositorio clonado:

```
cd nombre-proyecto/
```

7 Desvincular del remoto, lo repositorios clonados

Estando en la carpeta del repositorio que queremos desvincular

```
git remote remove origin
```

8 Ver Branch (todas ramas creadas)

Para ver las ramas creadas por nosotros u otros usuarios, podemos usar el siguiente comando:

git branch -a

```
anton@Antonio-DellG7 MINGW64 ~/proyecto-prueba (main)
$ git branch -a
* main
  remotes/origin/HEAD -> origin/main
  remotes/origin/main
```

La rama con el asterisco indica en la que estamos actualmente ubicados.

9 Actualizar nuestro repositorio local (actualiza la rama local)

Estando ubicados en la rama que queremos actualizar:

git fetch

git pull

```
anton@Antonio-DellG7 MINGW64 ~/proyecto-prueba (main)
$ git pull
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (1/1), done.
remote: Total 3 (delta 1), reused 3 (delta 1), pack-reused 0
Unpacking objects: 100% (3/3), 282 bytes | 23.00 KiB/s, done.
From https://github.com/ernestomedinam/proyecto-prueba
   c51d809..43d9a23  main       -> origin/main
Updating c51d809..43d9a23
Fast-forward
 index.html | 2 ++
 1 file changed, 2 insertions(+)
 create mode 100644 index.html
```

Tener en cuenta que el comando **git pull** hace es una especie de git fetch (sólo pide información al repositorio remoto, pero únicamente de la rama en la que nos ubicamos) con

una especie de git merge (sólo combina la información del repositorio remoto de la misma rama en la que estamos con el local, no hace merge con otras ramas).

10 Verificamos repositorios vinculados:

```
git remote -v
```

11 Operaciones comunes con ramas o Branchs:

11.1 Crear una nueva rama sin ubicarnos en ella

Para crear una nueva rama, hay que entender que se creará en base al último commit de la rama en la que nos encontramos, si nos encontramos en la rama **main** entonces sería el último commit de la rama **main**:

```
git branch nombre-nueva-rama
```

Si por el contrario, se quiere forzar la creación de una rama nueva basada en el último commit de la rama **main**, sin que estemos ubicados en la rama **main** (**se recomienda este método**):

```
git branch nombre-nueva-rama main
```

11.2 Ubicarse en una rama para trabajar en ella

```
git checkout nombre-de-la-rama-a-ir
```

11.3 Crear una nueva rama y ubicarse en ella automáticamente (sería la combinación de los pasos 11.1 y 11.2)

En esta forma de hacerlo, debemos entender que la copia que se hará en la nueva rama será de la rama en la que estamos ubicados actualmente.

```
git checkout -b nombre-nueva-rama
```

11.4 Crear una nueva rama y hacerle push directamente con modificaciones que tengamos

```
git push origin HEAD:nombre-de-la-rama-nueva
```

11.5 Eliminación de una rama (eliminación en el repositorio local)

El siguiente comando es una operación “segura”, ya que antes de eliminar la rama hace una comprobación que impide borrarla si aún no se han hecho merge de los cambios.

```
git branch -d nombre-de-la-rama
```

Si aún se desea eliminar la rama aunque no se haya hecho merge, se puede forzar su eliminación:

```
git branch -D nombre-de-la-rama
```

11.6 Eliminación de una rama en el repositorio remoto (GitHub)

Antes de aplicar los siguientes códigos, primero debe haberse hecho el paso 11.5

```
git push origin --delete nombre-de-la-rama
```

git push origin nombre-de-la-rama

12 Reset del repositorio local para que sea igual a una rama remota en GitHub (ejemplo con la rama main. Nota: peligro de perder los cambios locales)

git fetch origin

git reset --hard origin/main

13 Merge entre ramas

Para hacer merge entre ramas, debemos tener en cuenta dos cosas, la rama a la cuál vendrán todos los cambios de la rama principal, la cuál será **feature/login** para este ejemplo; y la rama principal, **main** para este ejemplo.

Lo primero es ubicarse en la rama **main**

git checkout main

Hacer fetch y pull por si hubo cambios

git fetch
git pull

Ahora vamos a la rama **feature/login**

git checkout feature/login

Estando en la rama **feature/login**, nos traemos los cambios de la rama **main** (merge local):

```
git merge main
```

Finalmente, actualizamos los cambios de la rama **feature/login** que ya tuvo el merge:

```
git add .
```

```
git commit -m "merge con main"
```

```
git push origin HEAD
```

Podemos verificar los cambios con un:

```
$ git log
```

Ahora desde la página de GitHub podemos realizar un pull request hacia la rama **main**, hay que tener cuidado con el sentido de las flechas en el momento de seleccionar las ramas involucradas para el pull request.

14 Flujo de trabajo cooperativo en repositorios.

Lo primero es evitar a toda costa trabajar directamente en la rama principal **main**. Por lo tanto, se debe crear una rama nueva a partir del último commit de la rama **main**.

Dos personas deben evitar a toda costa trabajar sobre la misma rama.

Una vez se tenga seguridad de los cambios y funcionalidades de una **rama bifurcada**, y previamente ya “pusheada” al repositorio con:

```
git add .
```

```
git commit -m “Comentario obligatorio”
```

```
git push origin HEAD
```

Se deben seguir los pasos del punto **13 Merge entre ramas**, para traerse los últimos cambios de la rama **main** a la rama bifurcada sobre la que se está trabajando.

Una vez revisado los **todos los conflictos** en la rama bifurcada, se procede a realizar un Pull Request a la rama **main** desde la página web del repositorio en github.