

GIT

Creando nuestro repositorio

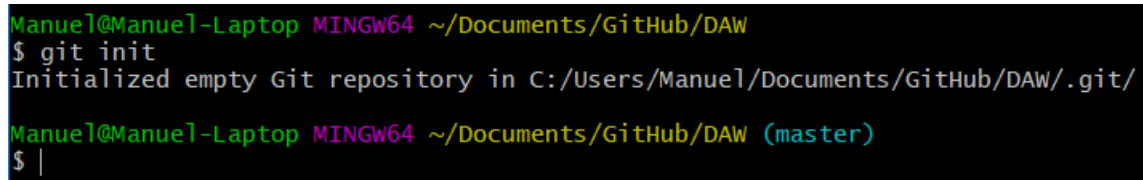
Para crear nuestro repositorio primero necesitaremos crear una carpeta en la que almacenaremos nuestro proyecto. Hacemos click derecho en la carpeta y seleccionamos la opción <<**Git Bash Here**>>.

Esto nos abrirá una consola de comandos que nos permitirá manipular la carpeta y gestionar nuestro repositorio Git.



```
MINGW64:/c/Users/Manuel/Documents/GitHub/DAW
Manuel@Manuel-Laptop MINGW64 ~/Documents/GitHub/DAW
$ |
```

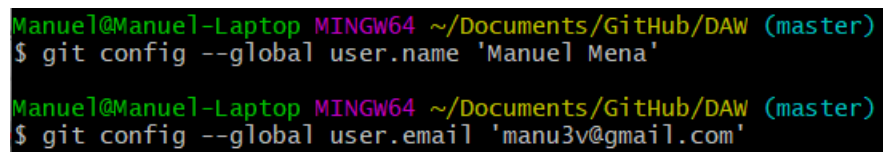
Después de esto vamos a proceder a crear nuestro repositorio. Para ello lo haremos con el comando <<**git init**>> desde la carpeta de nuestro proyecto.



```
Manuel@Manuel-Laptop MINGW64 ~/Documents/GitHub/DAW
$ git init
Initialized empty Git repository in C:/Users/Manuel/Documents/GitHub/DAW/.git/
Manuel@Manuel-Laptop MINGW64 ~/Documents/GitHub/DAW (master)
$ |
```

Este comando creará una carpeta invisible en la se almacenan archivos de configuración y de gestión del repositorio.

A continuación procederemos a añadir nuestro nombre y dirección de correo a Git. Lo haremos con estos comandos:



```
Manuel@Manuel-Laptop MINGW64 ~/Documents/GitHub/DAW (master)
$ git config --global user.name 'Manuel Mena'

Manuel@Manuel-Laptop MINGW64 ~/Documents/GitHub/DAW (master)
$ git config --global user.email 'manu3v@gmail.com'
```

Ya tenemos nuestro repositorio creado.

Creación, modificación y gestión de archivos

Podemos crear desde esta consola los archivos que tendría nuestro proyecto. Vamos a crear un <<index.html>> y un <<style.css>>. Haremos esto con el comando touch.

```
Manuel@Manuel-Laptop MINGW64 ~/Documents/GitHub/DAW (master)
$ touch index.html

Manuel@Manuel-Laptop MINGW64 ~/Documents/GitHub/DAW (master)
$ touch style.css
```

A continuación introducimos el comando <<git status>>. Este comando nos informará del estado de los archivos de nuestro repositorio respecto a la ultima versión almacenada en el repositorio. Nos mostrará aquellos archivos que han sido añadidos a la carpeta del proyecto pero que no han sido añadidos al repositorio, y a su vez, aquellos que hayan sufrido cambios pero que no se hayan subido al staging area.

```
Manuel@Manuel-Laptop MINGW64 ~/Documents/GitHub/DAW (master)
$ git status
On branch master

No commits yet

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

        index.html
        style.css

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

Como se puede ver en esta imagen, tenemos dos archivos dentro de la carpeta del repositorio pero estos no estan añadidos al repositorio. Para añadir elementos a nuestro repositorio lo haremos con el comando <<git add <file>...>>.

```
Manuel@Manuel-Laptop MINGW64 ~/Documents/GitHub/DAW (master)
$ git add index.html

Manuel@Manuel-Laptop MINGW64 ~/Documents/GitHub/DAW (master)
$ git status
On branch master

No commits yet

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

        new file:   index.html

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

        style.css
```

Al añadir el archivo y realizar un <<git status>> vemos como el archivo ha sido añadido a la staging area y esta listo para realizar un commit.

Si ahora quisiéramos eliminar un archivo del staging area lo haríamos con el siguiente comando:

```
Manuel@Manuel-Laptop MINGW64 ~/Documents/GitHub/DAW (master)
$ git rm --cached index.html
rm 'index.html'
```

Podemos comprobar que se ha eliminado de la staging area con el comando git status.

```
Manuel@Manuel-Laptop MINGW64 ~/Documents/GitHub/DAW (master)
$ git status
On branch master

No commits yet

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

        index.html
        style.css

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

Podemos añadir múltiples archivos con variaciones de Git add.

- **Git add .** Añade todos los archivos presentes en la carpeta del repositorio.
- **Git add *.html** Añade todos los archivos que tengan la extensión .html.

```
Manuel@Manuel-Laptop MINGW64 ~/Documents/GitHub/DAW (master)
$ git add .

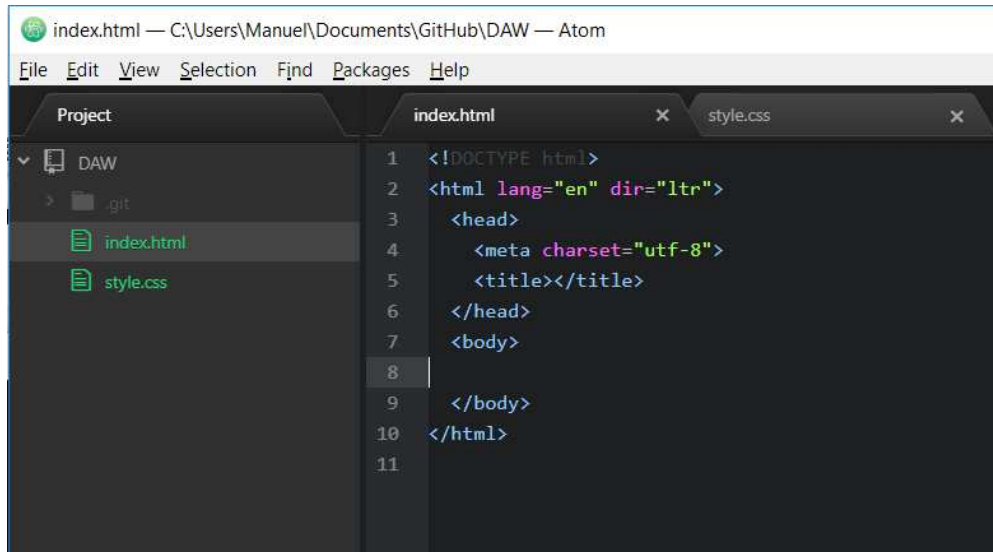
Manuel@Manuel-Laptop MINGW64 ~/Documents/GitHub/DAW (master)
$ git status
On branch master

No commits yet

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

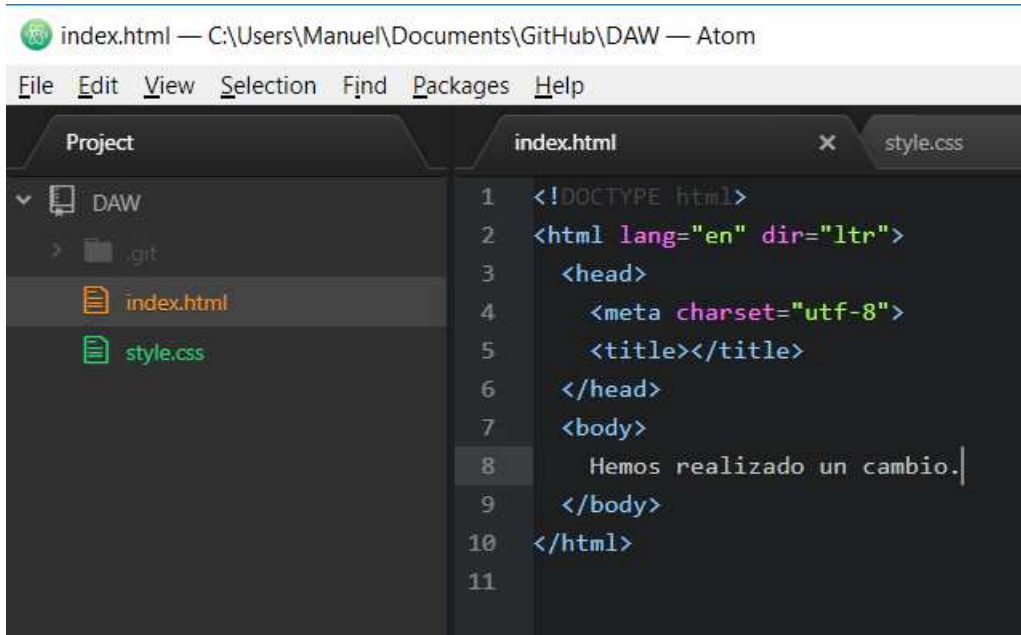
        new file:   index.html
        new file:   style.css
```

Si algún archivo recibiera un cambio, el comando git status nos informaría de que el archivo en la staging area ha recibido un cambio.



The screenshot shows the Atom editor interface. The title bar indicates the file is 'index.html' located at 'C:\Users\Manuel\Documents\GitHub\DAW'. The menu bar includes File, Edit, View, Selection, Find, Packages, and Help. The Project sidebar on the left shows a tree view with 'DAW' as the root, containing a '.git' folder, 'index.html', and 'style.css'. The main editor pane shows the content of 'index.html' with line numbers 1 through 11. The code is a basic HTML document structure:

```
1 <!DOCTYPE html>
2 <html lang="en" dir="ltr">
3   <head>
4     <meta charset="utf-8">
5     <title></title>
6   </head>
7   <body>
8
9   </body>
10 </html>
11
```



This screenshot shows the same Atom editor interface, but the file 'index.html' has been modified. The title bar and menu bar remain the same. In the Project sidebar, 'index.html' is now highlighted with an orange icon, indicating it has been changed. The main editor pane shows the updated content of 'index.html' with line numbers 1 through 11. The modification is on line 8, where the text 'Hemos realizado un cambio.' has been added inside the body tags:

```
1 <!DOCTYPE html>
2 <html lang="en" dir="ltr">
3   <head>
4     <meta charset="utf-8">
5     <title></title>
6   </head>
7   <body>
8     Hemos realizado un cambio.
9   </body>
10 </html>
11
```

```
Manuel@Manuel-Laptop MINGW64 ~/Documents/GitHub/DAW (master)
$ git status
On branch master

No commits yet

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

    new file:   index.html
    new file:   style.css

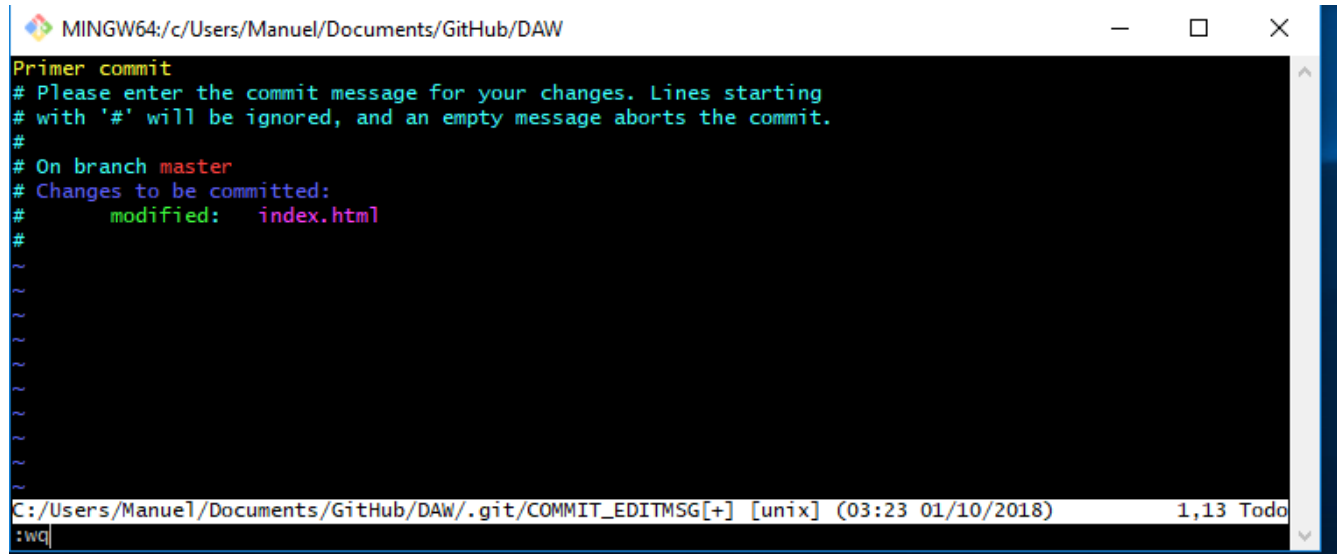
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   index.html
```

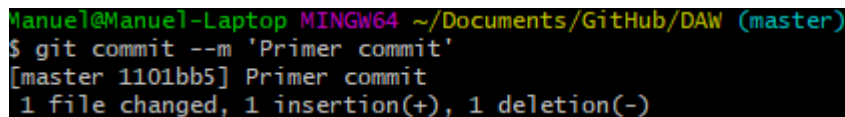
Hasta ahora no hemos añadido nada realmente a nuestro repositorio, simplemente lo hemos añadido a la staging area. Para añadir a nuestro repositorio los archivos de la staging area usaremos <<**git commit**>>.

Git commit nos abre un editor de texto VIM, en el que, para escribir tendremos que pulsar la tecla Insert, Escape una vez hayamos terminado de escribir, y :wq para cerrar el editor y guardar los cambios.

En este editor de texto escribiremos una pequeña descripción de los cambios realizados respecto a la anterior versión del repositorio.

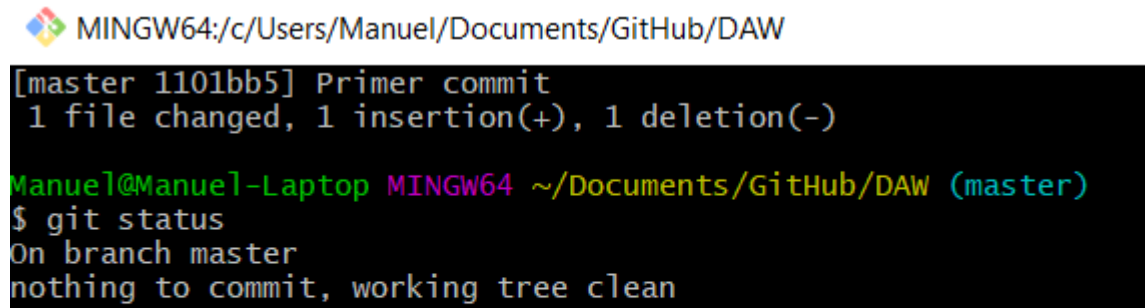


```
MINGW64:/c/Users/Manuel/Documents/GitHub/DAW
Primer commit
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch master
# Changes to be committed:
#   modified:   index.html
#
~
~
~
~
~
~
~
~
~
~
C:/Users/Manuel/Documents/GitHub/DAW/.git/COMMIT_EDITMSG[+] [unix] (03:23 01/10/2018) 1,13 Todo
:wq
```



```
Manuel@Manuel-Laptop MINGW64 ~/Documents/GitHub/DAW (master)
$ git commit --m 'Primer commit'
[master 1101bb5] Primer commit
1 file changed, 1 insertion(+), 1 deletion(-)
```

Podemos saltarnos tener que interactuar con el editor de texto escribiendo en lugar de git commit, <<**git commit -m 'descripcion'**>>



```
MINGW64:/c/Users/Manuel/Documents/GitHub/DAW
[master 1101bb5] Primer commit
1 file changed, 1 insertion(+), 1 deletion(-)

Manuel@Manuel-Laptop MINGW64 ~/Documents/GitHub/DAW (master)
$ git status
On branch master
nothing to commit, working tree clean
```

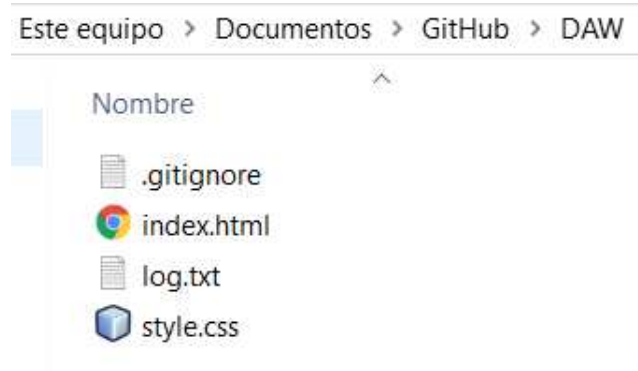
.gitignore

Para filtrar aquellos archivos/carpetas dentro de nuestro proyecto que no queramos que el repositorio incluya, podemos crear el archivo .gitignore. Este archivo sera una lista con los nombres de los archivos que nuestro repositorio no incluirá.

```
MINGW64:/c/Users/Manuel/Documents/GitHub/DAW

Manuel@Manuel-Laptop MINGW64 ~/Documents/GitHub/DAW (master)
$ touch .gitignore

Manuel@Manuel-Laptop MINGW64 ~/Documents/GitHub/DAW (master)
$ touch log.txt
```



Tenemos nuestro archivo .gitignore, y un archivo auxiliar que no queremos incluir (log.txt). Como el archivo .gitignore se encuentra vacío, si hacemos git status, veremos tanto .gitignore como log.txt.

```
MINGW64:/c/Users/Manuel/Documents/GitHub/DAW

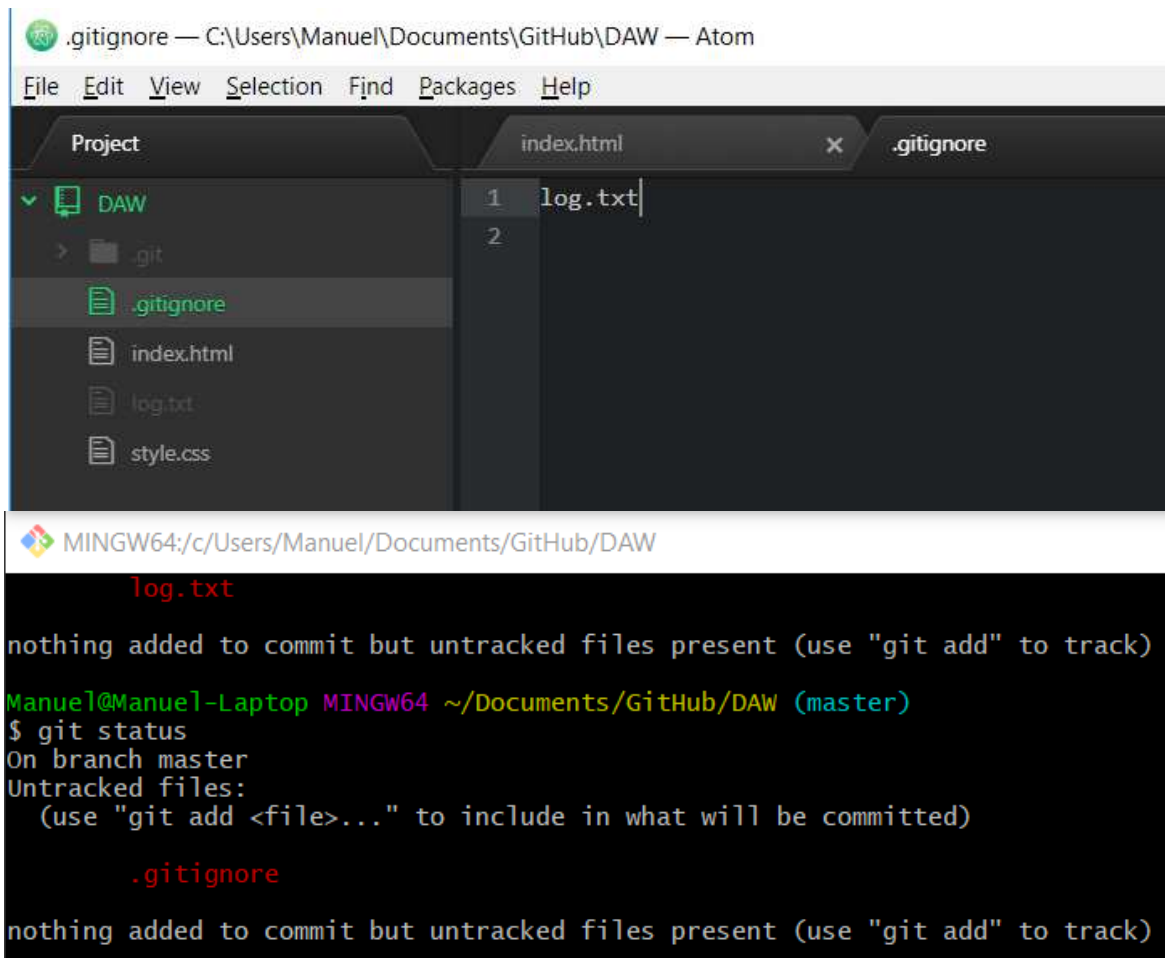
Manuel@Manuel-Laptop MINGW64 ~/Documents/GitHub/DAW (master)
$ touch log.txt

Manuel@Manuel-Laptop MINGW64 ~/Documents/GitHub/DAW (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .gitignore
        log.txt

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

Pero en el momento en el que añadimos log.txt al archivo .gitignore, git dejara de reconocerlo como un archivo a incluir.



The image shows two windows. The top window is the Atom text editor, titled ".gitignore — C:\Users\Manuel\Documents\GitHub\DAW — Atom". It has a menu bar with File, Edit, View, Selection, Find, Packages, and Help. The left sidebar shows a project tree for "DAW" with sub-items ".git", ".gitignore", "index.html", "log.txt", and "style.css". The main editor area shows the ".gitignore" file with two lines: "1 log.txt" and "2". The bottom window is a terminal titled "MINGW64:/c/Users/Manuel/Documents/GitHub/DAW". It shows the following text:

```
log.txt
nothing added to commit but untracked files present (use "git add" to track)
Manuel@Manuel-Laptop MINGW64 ~/Documents/GitHub/DAW (master)
$ git status
On branch master
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)
```

Este proceso funciona de la misma manera con directorios enteros añadiendo el nombre del directorio (/nombredirectorio) a la lista de .gitignore.

Branches

Las ramas o branches nos permiten mantener diferentes versiones de un proyecto que durante algun momento de su desarrollo se han distinguido de la rama “principal” (master) de desarrollo.

Para crear una nueva rama escribimos “**git branch nombrerama**”.

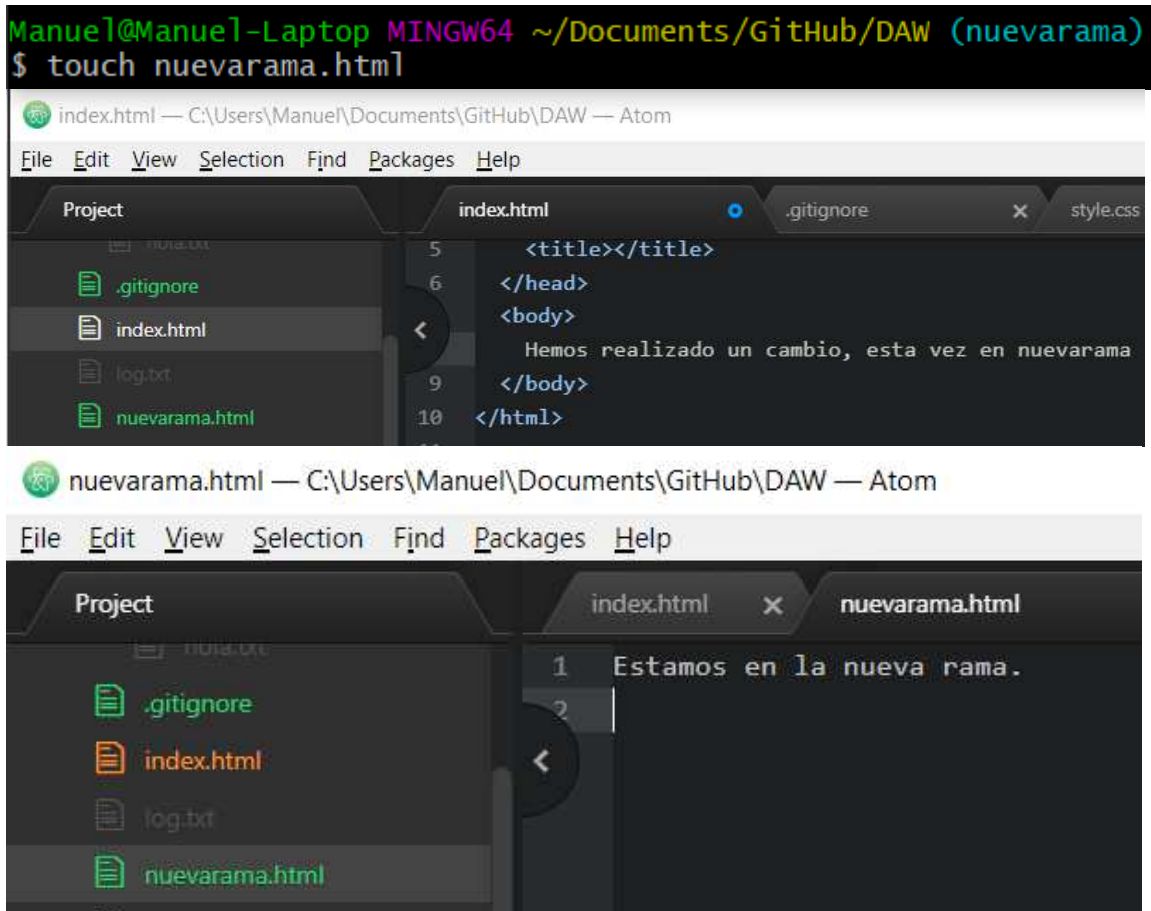
```
Manuel@Manuel-Laptop MINGW64 ~/Documents/GitHub/DAW (master)
$ git branch nuevarama
```

Haciendo esto solo nos creara la rama, pero no nos cambiara a ella. Para que nos cambie a esa rama tendremos que introducir “**git checkout nombrerama**”.

```
Manuel@Manuel-Laptop MINGW64 ~/Documents/GitHub/DAW (master)
$ git checkout nuevarama

Switched to branch 'nuevarama'
```

Una vez en esta rama lo que añadamos, borremos o modifiquemos se aplicara unicamente a esta rama.




```
Manuel@Manuel-Laptop MINGW64 ~/Documents/GitHub/DAW (nuevarama)
$ git status
On branch nuevarama
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   index.html

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

        .gitignore
        nuevarama.html

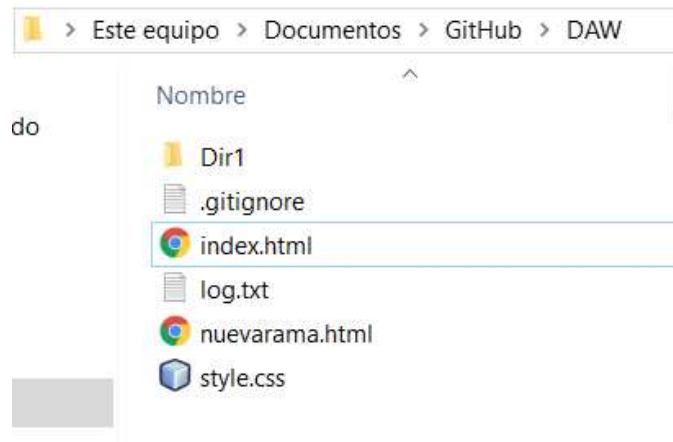
no changes added to commit (use "git add" and/or "git commit -a")
```

Añadimos estos archivos nuevos, y modificados a la staging area y realizamos un commit.

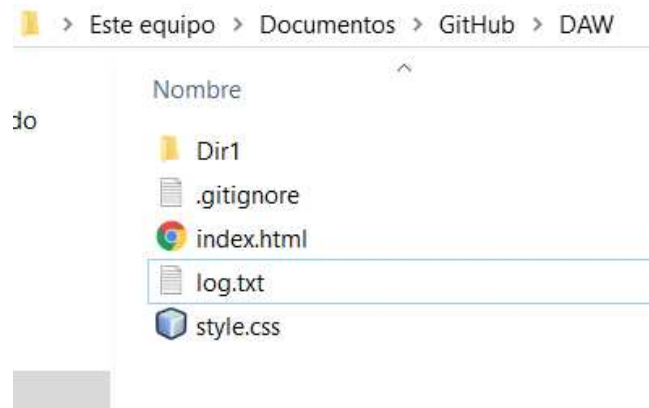
```
Manuel@Manuel-Laptop MINGW64 ~/Documents/GitHub/DAW (nuevarama)
$ git commit -m 'Primer commit en nuevarama'
[nuevarama 7db21c2] Primer commit en nuevarama
3 files changed, 4 insertions(+), 1 deletion(-)
create mode 100644 .gitignore
create mode 100644 nuevarama.html
```

Ahora disponemos de dos ramas (master y nuevarama), con archivos distintos en ambas. De hecho si volviésemos a la rama master, en el explorador de Windows veríamos cambiar los archivos de la carpeta del proyecto dependiendo de en que rama estemos.

Nuevarama:



Master:



Merge

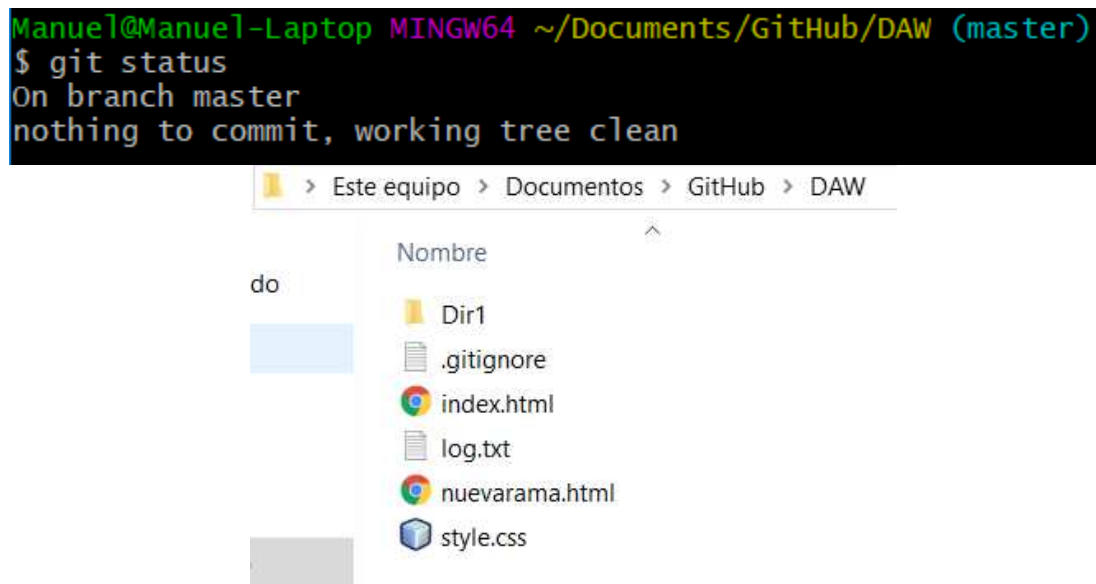
Cuando queremos unir dos ramas distintas en una misma utilizaremos “git merge nombrerama” desde la rama principal (master).

```
Manuel@Manuel-Laptop MINGW64 ~/Documents/GitHub/DAW (master)
$ git merge nuevarama
```

Al hacer este comando se abre el editor de texto VIM. Introducimos una descripción de la necesidad de este merge.

```
Merge branch 'nuevarama'
Merge con nuevarama
Manuel@Manuel-Laptop MINGW64 ~/Documents/GitHub/DAW (master)
$ git merge nuevarama
Merge made by the 'recursive' strategy.
index.html      | 2 +-
nuevarama.html  | 1 +
2 files changed, 2 insertions(+), 1 deletion(-)
create mode 100644 nuevarama.html
~
```

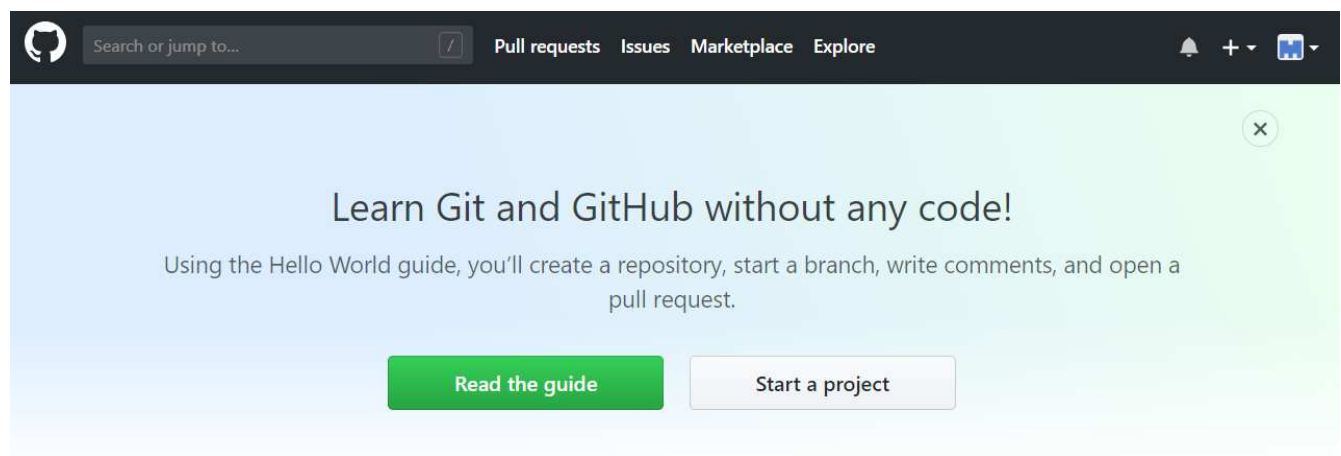
Ahora, desde el explorador de Windows, vemos que aunque estamos en la rama master, tenemos los archivos de la rama nuevarama.



Repositorio remoto

Para crear tu repositorio remoto, iremos a la pagina <https://github.com/>, nos crearemos una cuenta si no disponemos ya de una.

Una vez logeado en GitHub le damos a “**Start a project**”.




Introducimos los datos del nuevo repositorio en GitHub.

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner

Repository name

 ManuelMenaM ▾


 /

DAW - Desafio1 ✓


Great repository names are short, lowercase, and contain only numbers, letters, hyphens, and underscores. Your new repository will be created as DAW---Desafio1 wing-octo-spoon.

Description (optional)

Desafio 1 de Despliegue Aplicaciones Web

☒  **Public**

Anyone can see this repository. You choose who can commit.


☐  **Private**

You choose who can see and commit to this repository.

☐ **Initialize this repository with a README**

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None ▾

Add a license: None ▾ 

Create repository

Te dan dos opciones de privacidad para tu repositorio:

- **Public.** Todo el mundo podrá ver tu repositorio, pero no podrán editar. Tendrán que pedir permiso para poder editar.
- **Private.** Solo podrán ver y editar tu repositorio aquellos que reciban tu autorización. Es de pago.

También se te da la opción de añadir un archivo README, pero es opcional.

Le damos a **Create repository**

En la consola de git de nuestro repositorio introducimos estos comandos:

- **git remote add origin https://github.com/ManuelMenaM/DAW---Desafio1.git**
- **git push -u origin master.**

```
Manuel@Manuel-Laptop MINGW64 ~/Documents/GitHub/DAW (master)
$ git remote add origin https://github.com/ManuelMenaM/DAW---Desafio1.git

Manuel@Manuel-Laptop MINGW64 ~/Documents/GitHub/DAW (master)
$ git push -u origin master
Enumerating objects: 15, done.
Counting objects: 100% (15/15), done.
Delta compression using up to 4 threads
Compressing objects: 100% (12/12), done.
Writing objects: 100% (15/15), 1.51 KiB | 193.00 KiB/s, done.
Total 15 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), done.
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:   https://github.com/ManuelMenaM/DAW---Desafio1/pull/new/master
remote:
To https://github.com/ManuelMenaM/DAW---Desafio1.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

ManuelMenaM / DAW---Desafio1

Watch

0

Star

0

Fork

0

<> Code

Issues 0

Pull requests 0

Projects 0

Wiki

Insights

Settings

Desafio 1 de Despliegue Aplicaciones Web

Edit

Manage topics

5 commits

1 branch

0 releases

1 contributor

Branch: master

New pull request

Create new file

Upload files

Find file

Clone or download

ManuelMenaM Merge branch 'nuevarama'

Latest commit 4d02334 23 hours ago

.gitignore

Primer commit en nuevarama

a day ago

index.html

Primer commit en nuevarama

a day ago

nuevarama.html

Primer commit en nuevarama

a day ago

style.css

Primer Commit

a day ago

Help people interested in this repository understand your project by adding a README.

Add a README

Podemos añadir colaboradores desde la pestaña **Settings/Colaborators**, introduciendo el nombre de usuario, email, o el nombre completo del colaborador.

The screenshot shows the GitHub repository settings for 'ManuelMenaM / DAW---Desafio1'. The 'Collaborators' tab is active. The page displays a search bar to add collaborators and a list of existing collaborators (currently empty). The search bar is labeled 'Search by username, full name or email address' and includes a note: 'You'll only be able to find a GitHub user by their email address if they've chosen to list it publicly. Otherwise, use their username instead.' The 'Add collaborator' button is visible next to the search bar.

Modificación de un repositorio ajeno.

Necesitamos que el administrador del repositorio te envíe una invitación. Tras aceptar dicha invitación, desde la URL del repositorio compartido le damos a **Clone or Download**.

The screenshot shows the GitHub repository page for 'HannaDAW2 / Proyecto'. The 'Clone or download' button is highlighted, and a dropdown menu is open showing the 'Clone with HTTPS' option and the repository URL: `https://github.com/HannaDAW2/Proyecto.git`. The dropdown menu also includes a 'Use SSH' link and a 'Download ZIP' button. The repository page shows 3 commits, 1 branch, 0 releases, and 1 contributor.

Copiamos la URL que nos indica y la introducimos en la consola de Git tras el comando “git clone”

- **git clone** <https://github.com/nombreusuario/nombrerrepositorio.git>

```
Manuel@Manuel-Laptop MINGW64 ~/Documents/GitHub
$ git clone https://github.com/HannaDAW2/Proyecto.git
Cloning into 'Proyecto'...
remote: Enumerating objects: 29, done.
remote: Counting objects: 100% (29/29), done.
remote: Compressing objects: 100% (20/20), done.
remote: Total 29 (delta 5), reused 29 (delta 5), pack-reused 0
Unpacking objects: 100% (29/29), done.
```

Una vez en la carpeta del proyecto compartido, realizamos los cambios y hacemos los siguientes comandos:

- **git add .**
- **Git commit -m ‘descripción’.**
- **Git push**

```
Manuel@Manuel-Laptop MINGW64 ~/Documents/GitHub/Proyecto (master)
$ git add .

Manuel@Manuel-Laptop MINGW64 ~/Documents/GitHub/Proyecto (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   mejoradeManuel.txt

Manuel@Manuel-Laptop MINGW64 ~/Documents/GitHub/Proyecto (master)
$ git commit -m 'Mejora de Manuel'
[master 5f06958] Mejora de Manuel
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 mejoradeManuel.txt

Manuel@Manuel-Laptop MINGW64 ~/Documents/GitHub/Proyecto (master)
$ git push

Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 320 bytes | 320.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/HannaDAW2/Proyecto.git
 6b619bc..5f06958 master -> master
```


Aquí podemos ver que el cambio se ha realizado:

HannaDAW2 / Proyecto

Watch

0

Star

0

Fork

0

Code

Issues 0

Pull requests 0

Projects 0

Wiki

Insights

No description, website, or topics provided.

4 commits

1 branch

0 releases

1 contributor

Branch: master

New pull request

Create new file

Upload files

Find file

Clone or download

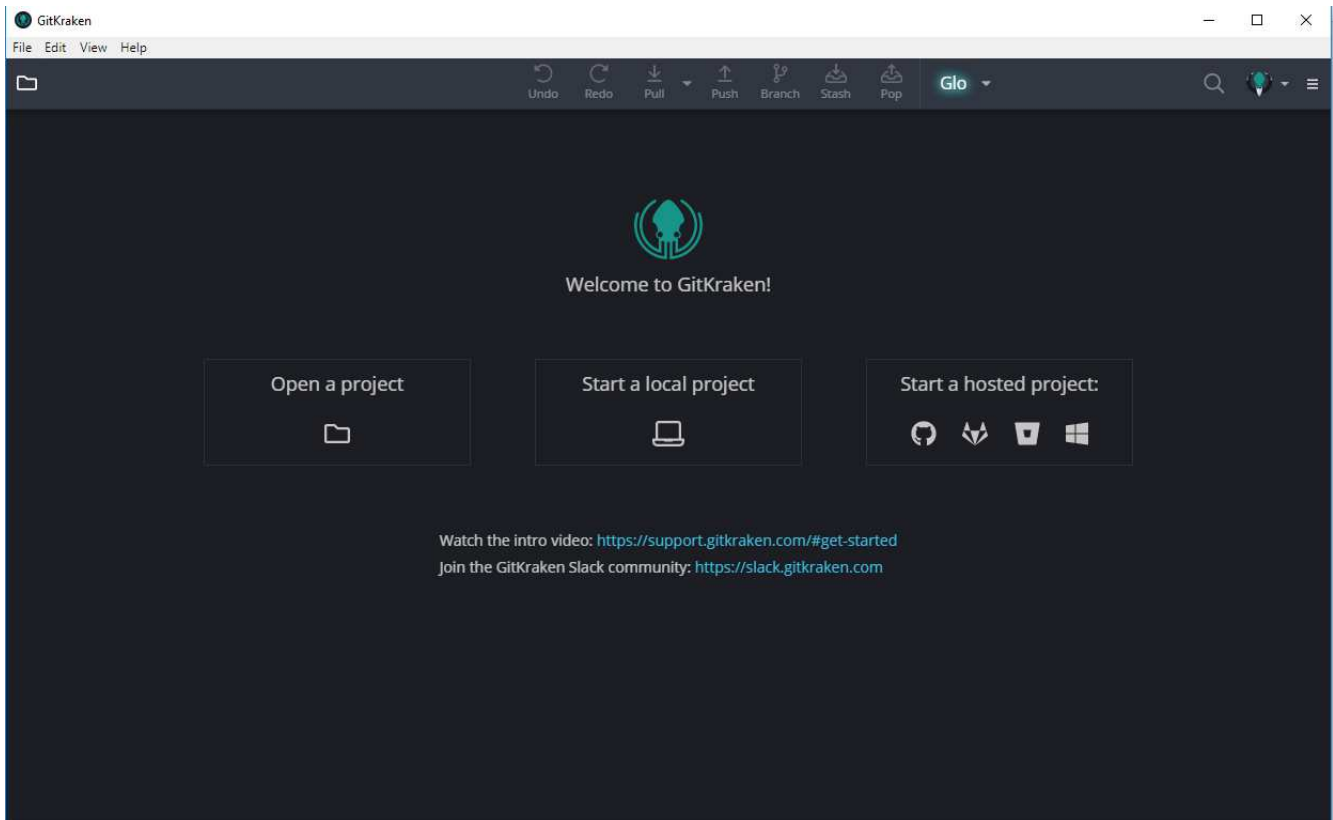
ManuelMenaM	Mejora de Manuel	Latest commit 5f06958 6 minutes ago
Desafio	Segundo commit	6 days ago
.gitignore	Nuria ha estado aqui	a day ago
mejoradeManuel.txt	Mejora de Manuel	6 minutes ago

Help people interested in this repository understand your project by adding a README.

Add a README

GitKraken

GitKraken es un entorno gráfico para Git. Nos permite la misma funcionalidad que al consola de Git, pero de forma gráfica y más intuitiva.



Dándole al icono de la carpeta situado en la esquina superior izquierda, GitKraken nos da la opción de iniciar un proyecto nuevo (local o remoto), abrir un proyecto local ya presente en tu ordenador, o clonar uno alojado en GitHub u otros servicios similares.

Repository Management

Open

Clone

Init

Open a Repository

Recently Opened

Favorites

Project Groups

Recently Opened

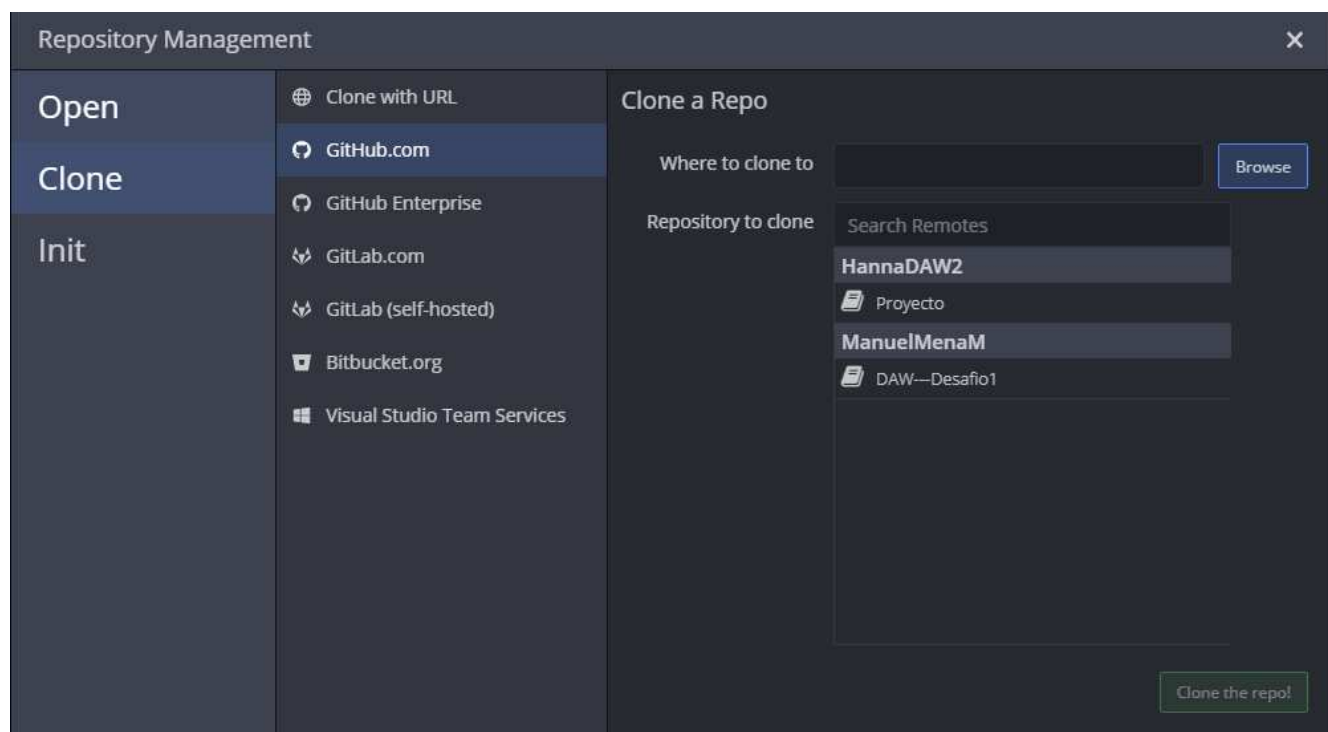
DAW

C:\Users\Manuel\Documents\GitHub\DAW

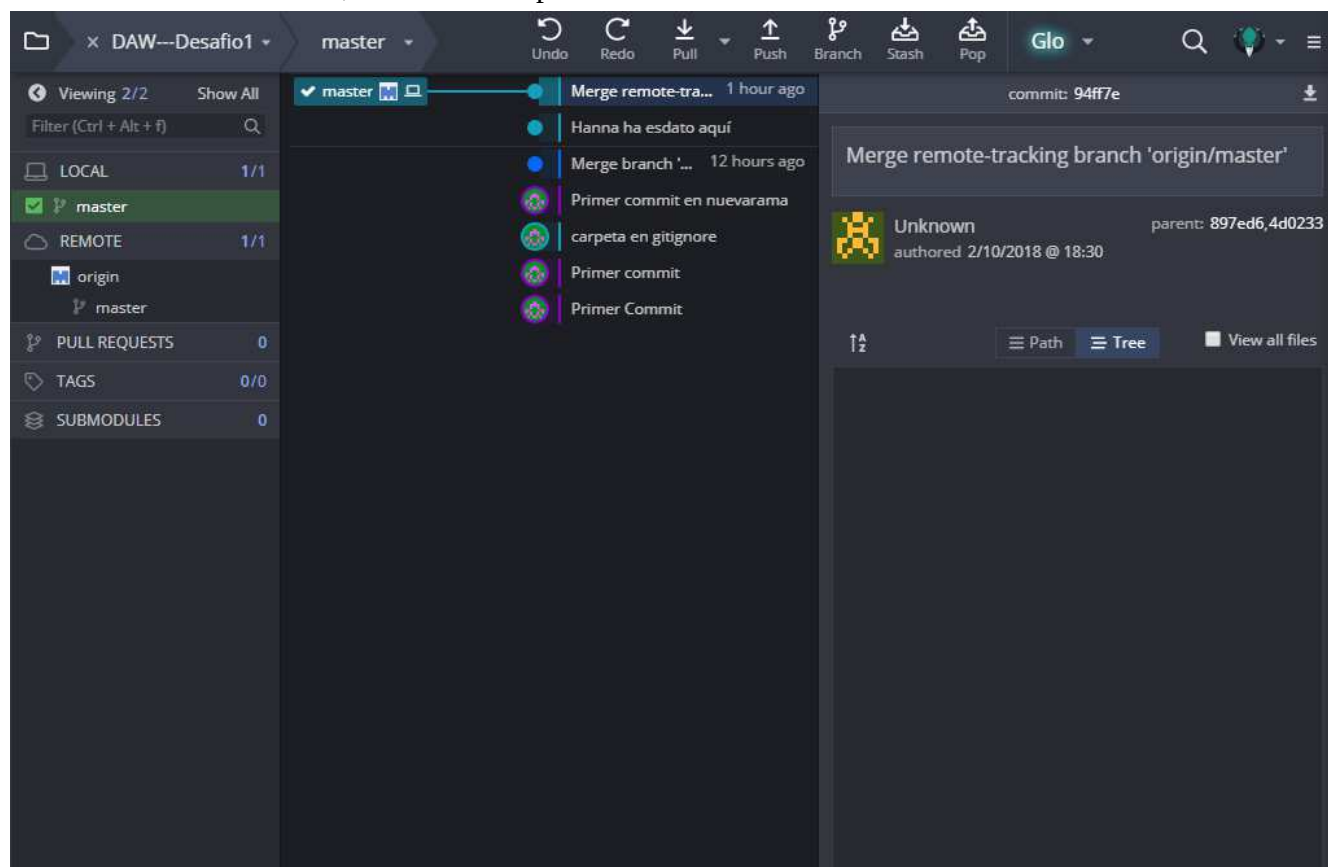
DIW

C:\Users\Manuel\Documents\GitHub\DIW

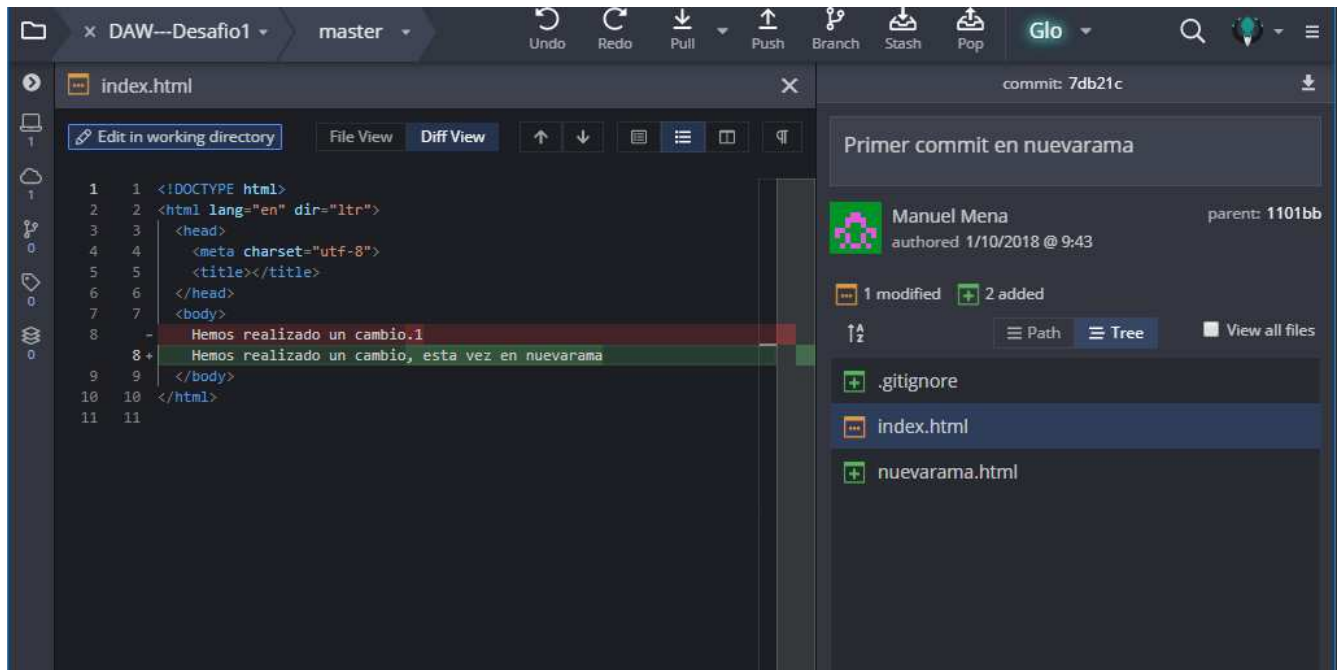
Si estamos vinculados a nuestra cuenta de GitHub, podremos ver directamente los proyectos a los que tenemos acceso a la hora de clonar.



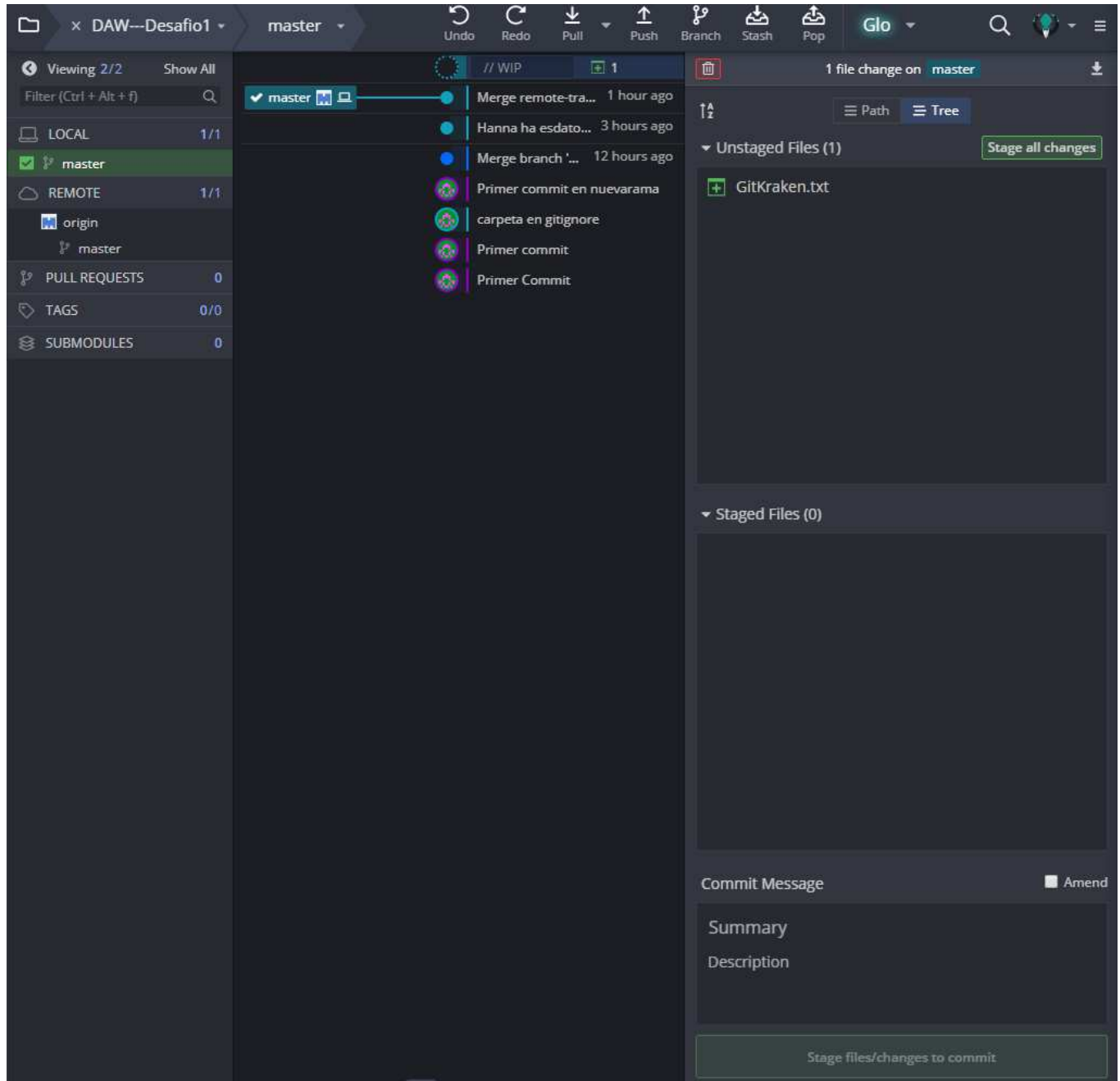
Una vez creado o clonado, GitKraken se presentara de esta forma o similar:



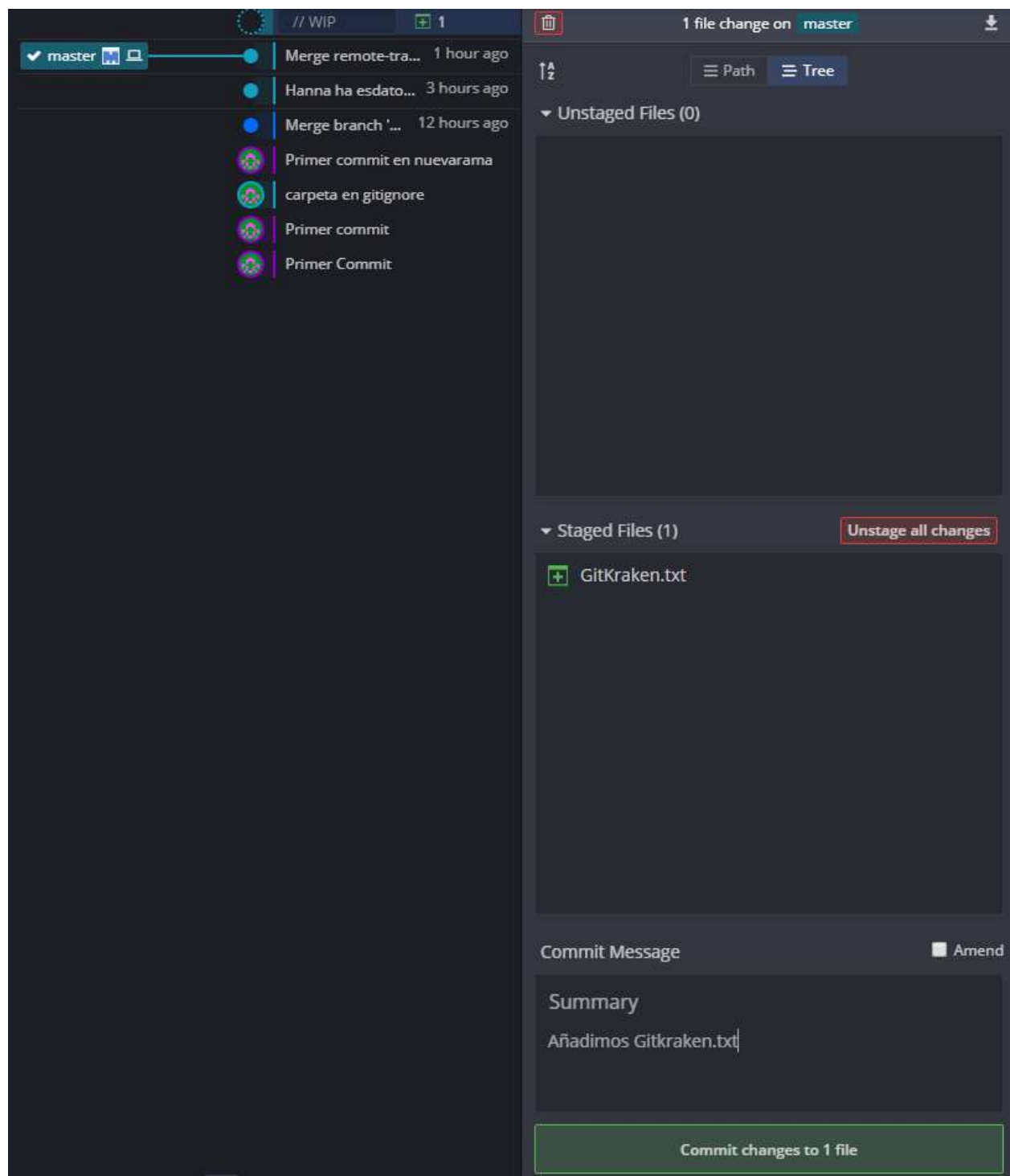
En el centro tenemos un historial de los merges, y commits realizados en este proyecto. Si pinchamos en cada entrada del historial en la zona de la derecha se nos mostrarán los archivos modificados en ese commit. Si además hacemos doble click en esos archivos, veremos los cambios realizados línea por línea.



Al añadir un archivo o modificar algún archivo existente, se abrirá una entrada en el historial que sera la de nuestro trabajo actual. Todos los cambios que realicemos o archivos nuevos que añadamos aparecerán en la zona de la derecha en la que se gestionarán los archivos que están en la staging area y los que no, a la vez que el mensaje de commit en caso de que lo hagamos.

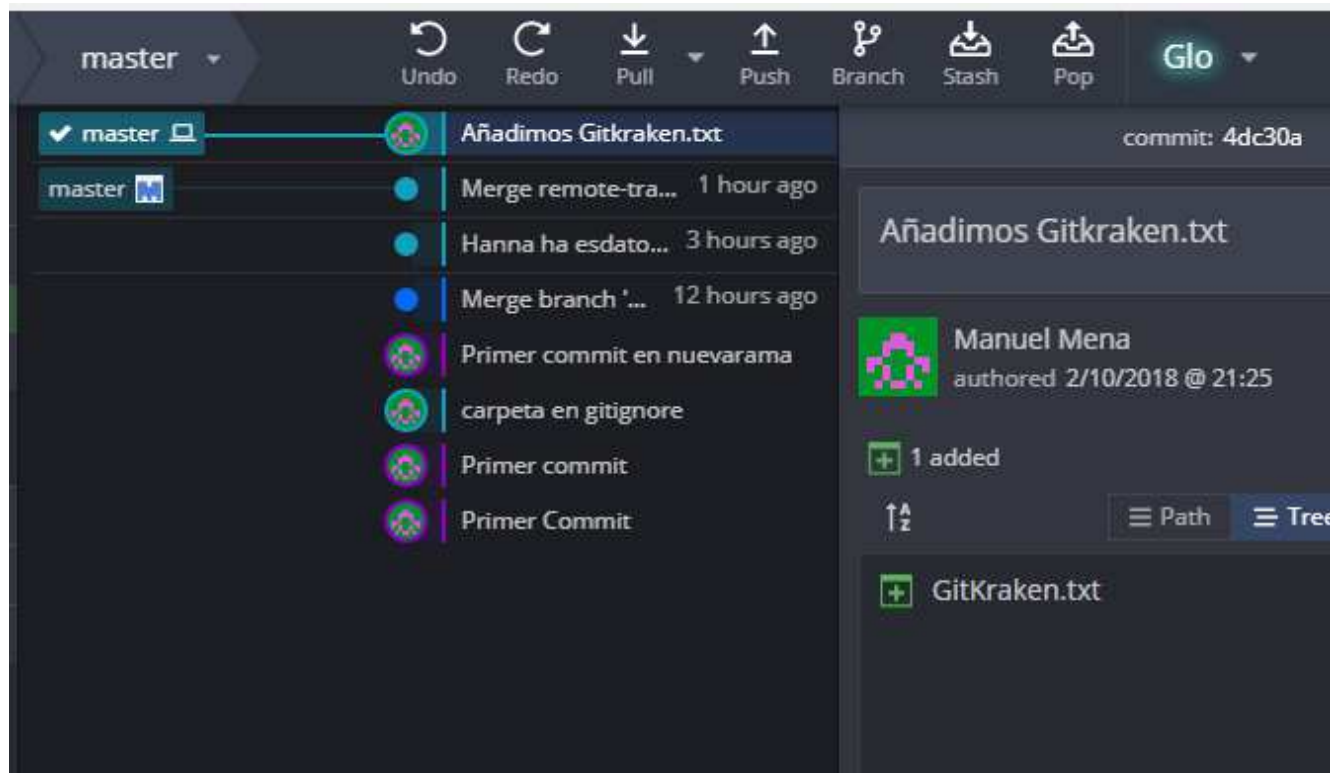


Se pueden añadir los archivos a la staging area dándole a “Stage all changes”, o se puede hacer por cada archivo individualmente haciendo click en “stage file” que aparecerá al poner el cursor sobre el.

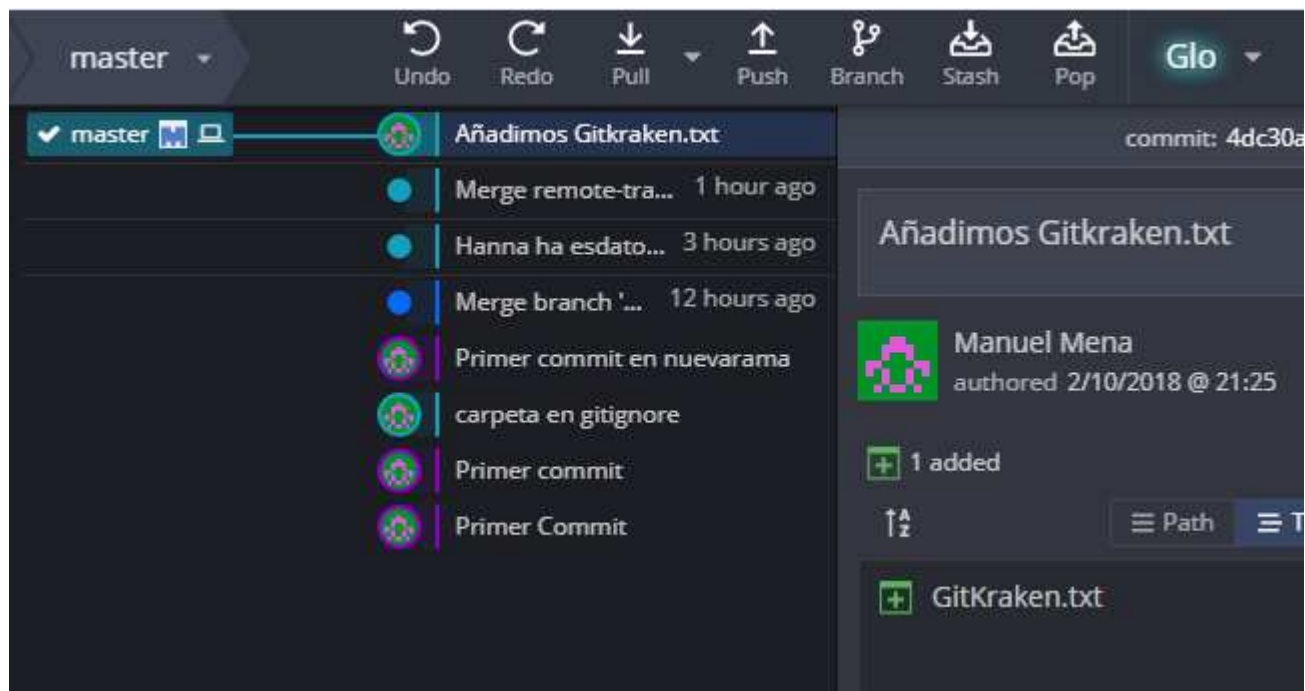


Una vez añadidos los archivos a la Staging area, se podrá hacer un commit.

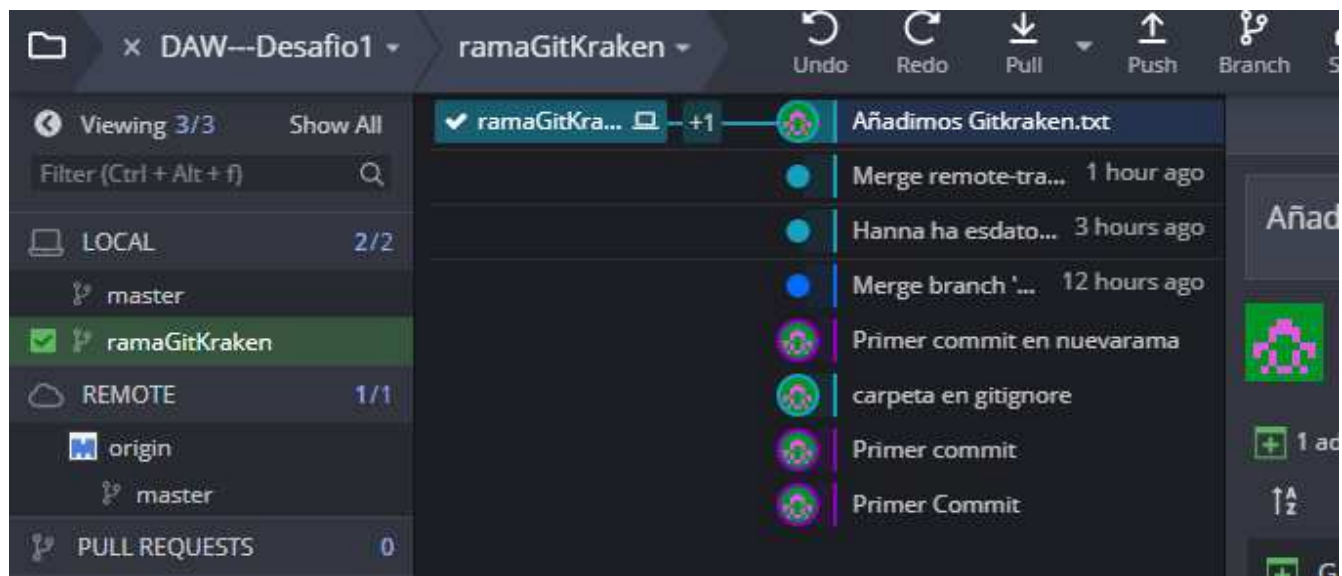
Podremos realizar el Push con simplemente pulsar el boton “Push” de la barra de botones situada en la parte superior de la interfaz.



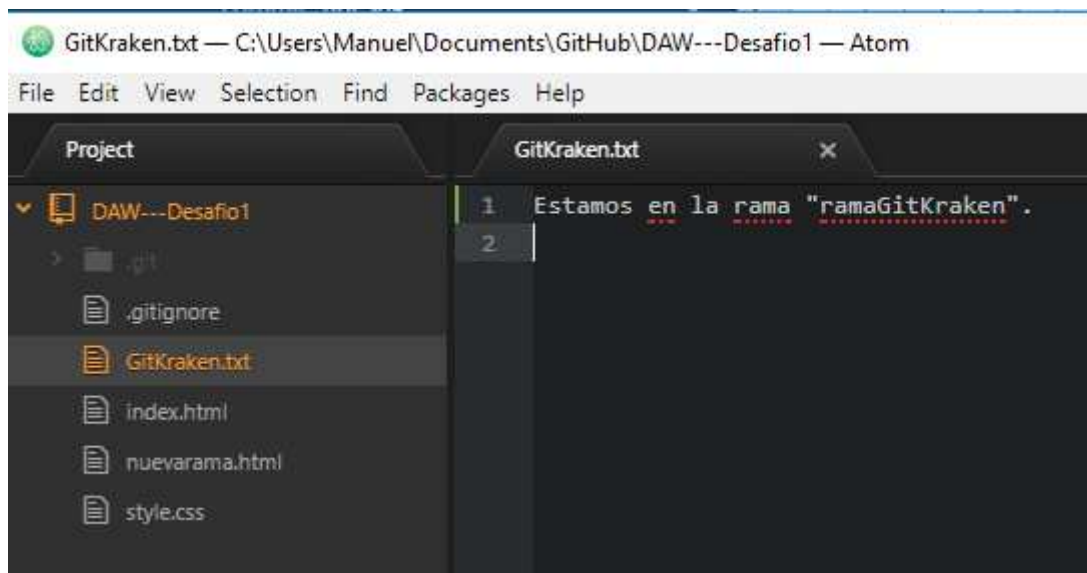
Como podemos ver, la rama master del repositorio remoto no es la misma que la presente en el repositorio local, tras hacer el Push, tanto el repositorio local, como el remoto estan en la misma versión.



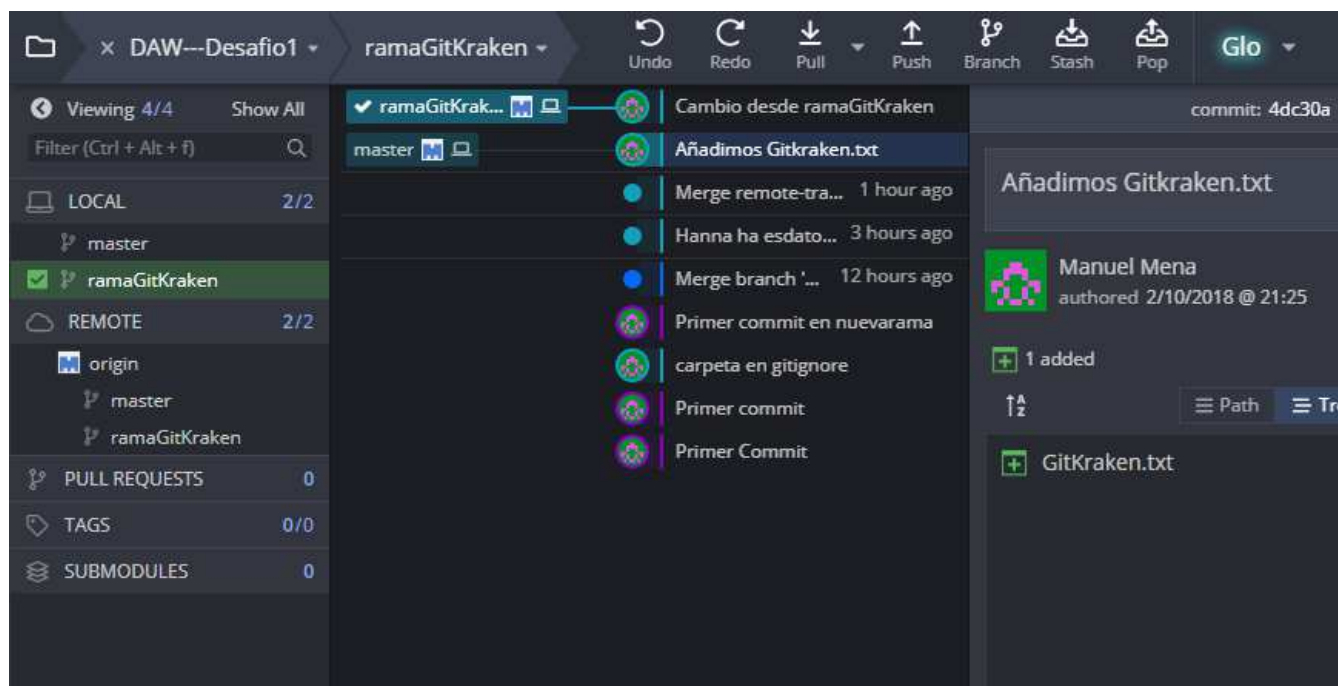
Dándole al botón Branch e introduciendo un nombre para la rama, crearemos una rama paralela a la rama master.



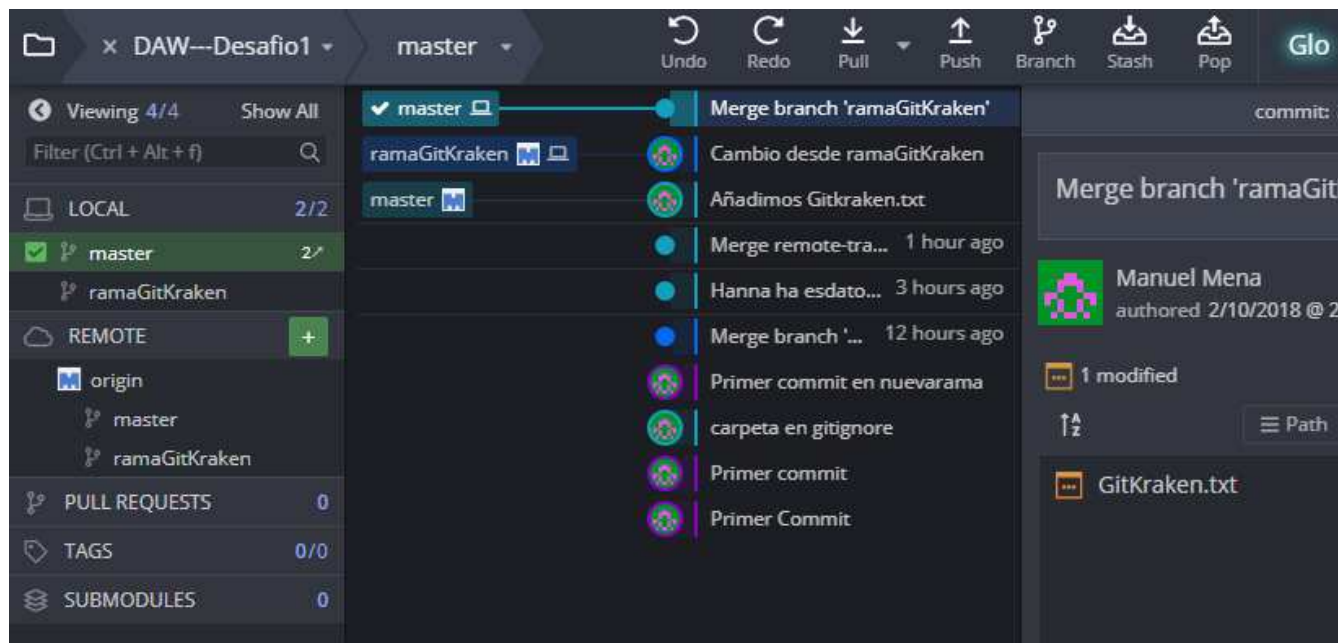
Vamos a editar el archivos GitKraken.txt mientras estamos en la rama “ramaGitKraken”.



Ahora, desde GitKraken, hacemos el commit y el push de este cambio, pasando a tener dos ramas diferentes (master y ramaGitKraken).



Para hacer un merge de ramaGitKraken a master simplemente arrastramos ramaGitKraken a la rama master y seleccionamos “Merge ramaGitKraken into master”.



Tras cambiar a la rama master y realizamos el Push dándole al boton “Push”. Con esto conseguimos que nuestro repositorio remoto esté sincronizado con nuestro repositorio local.

