



El problema

Git - El problema

Cuando trabajamos en un proyecto es necesario hacer **cambios** de manera constante, ya sea para **agregar funcionalidad**, cambiar el diseño, **arreglar bugs**, o lo que sea.

Sin importar la razón de los **cambios**, es probable que en un futuro tengamos que hacer que el proyecto se vuelva a comportar como antes, efectivamente anulando los cambios.

También es probable que tengamos que partir de un **cambio** intermedio (dos versiones atrás, por ejemplo) y en base a eso crear una **nueva versión**.

Éstos son sólo **ejemplos**, pero existen mil situaciones en las cuales es necesario poder acceder a **versiones anteriores** de nuestros proyectos.

Git - El problema

Una forma de resolverlo sería creando un **duplicado manual** para cada cambio, algo bastante normal en el mundo del diseño gráfico o de la escritura.



GIT



Otra forma un poco más ordenada sería la de utilizar una **herramienta** que se encargue de **gestionar estas versiones**, y que nos de una forma de viajar entre ellas como si fuera una máquina del tiempo.

Es ahí donde entra **Git**.



Git - Diferencias con Github

Git es la herramienta de control de versiones más usada y más conocida. Fue creada por el Linus Torvalds (el mismo que creó Linux) en el 2005 y está activamente en desarrollo al día de hoy.



GitHub, en cambio, es simplemente un lugar donde podemos alojar proyectos gestionados con Git. Análogamente, podemos pensar en GitHub como si fuera Google Drive o DropBox, ambos lugares dónde podemos guardar archivos.



Desarrollo distribuido y offline

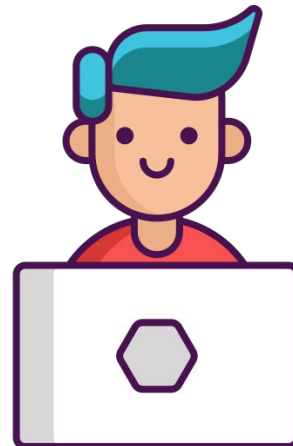
Con git, cada desarrollador que trabaje en un proyecto tiene una copia local no sólo del código sino de la historia del código, es decir de cada cambio. Esto nos permite trabajar de manera distribuida en tantos dispositivos como sea necesario e incluso sin conexión.

Luego con repositorios remotos el equipo puede sincronizar su trabajo según sus necesidades.



Nos ayuda a ser ordenados

Al obligarnos a subir cambios “de a poco”, terminamos creando una historia de código más ordenada y fácil de manejar.



Git - ¿Cómo lo instalo?

¡Es probable que ya lo tengan instalado! Git viene pre-instalado por defecto en Linux y en la mayoría de las Mac. En Windows no, pero suele venir empaquetado junto al Visual Studio Code y otros editores de texto.



Git - ¿Cómo lo instalo?

¿Cómo veo si ya lo tengo?

La forma más fácil es abrir una consola o terminal y escribir “git --version”.

A terminal window with a black background and three colored window control buttons (red, yellow, green) in the top left corner. The text "\$ git --version" is displayed in a monospaced font, with the dollar sign and "git" in orange and "--version" in green.

```
$ git --version
```

¿Y si no lo tengo?

Pueden descargarlo directamente de su sitio oficial:
<https://git-scm.com/>

Git - Configurar GIT

Lo primero que deberás hacer cuando instales Git es **establecer tu nombre de usuario y dirección de correo electrónico**. Esto es importante porque los "**commits**" de Git usan esta información, y es introducida de manera inmutable en los commits que envías:



```
$ git config --global user.name "John Doe"  
$ git config --global user.email johndoe@example.com
```

Git - ¿Cómo lo uso?

Hay varias maneras de utilizar **Git**, la más completa es hacerlo desde una **terminal**.

Git es una aplicación de línea de comandos (CLI), por lo que la mejor forma de utilizar todas su capacidades es a través de **comandos** (aunque cueste un poco al principio).

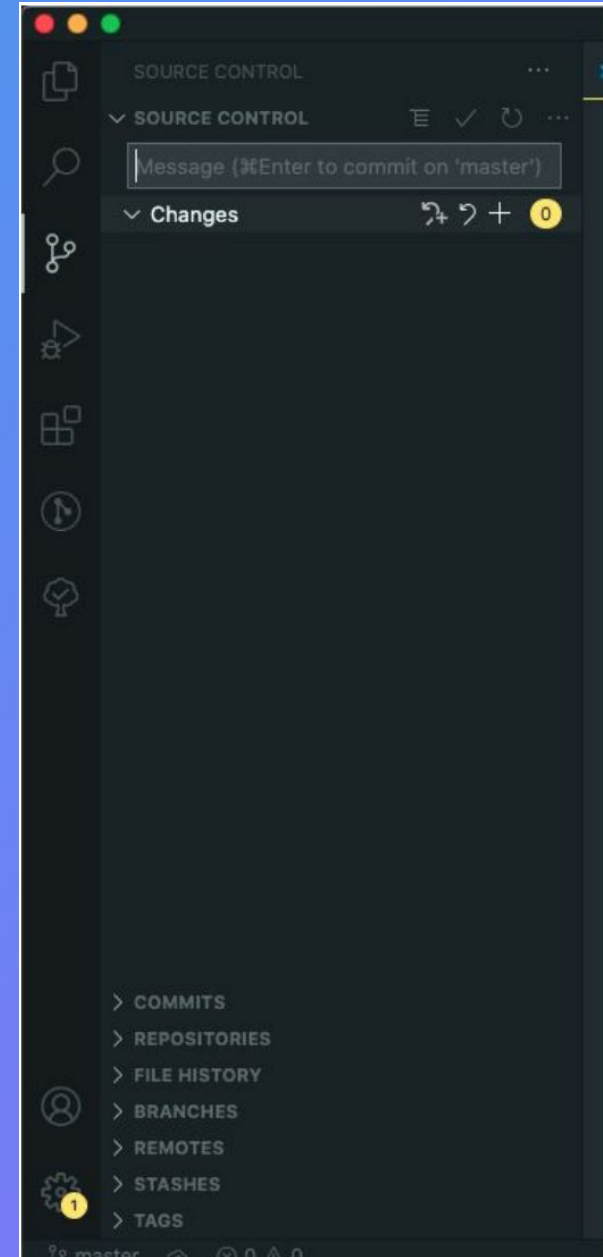
```
$ git init
Inicializado repositorio Git vacío en /Users/gustavguez/Workspace/Senpai Academy/FSD8/proyecto-ejemplo/.git/

$ touch README.md
$ git add .
$ git commit -m "Agregar README.md"
[main (commit-raíz) 5fddefa] Agregar README.md
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 README.md
```

Git - ¿Cómo lo uso?

Otra forma es hacerlo desde **VS Code**, utilizando la pestaña de Source Control.

Esta es la forma más fácil para empezar y la que vamos a utilizar por ahora



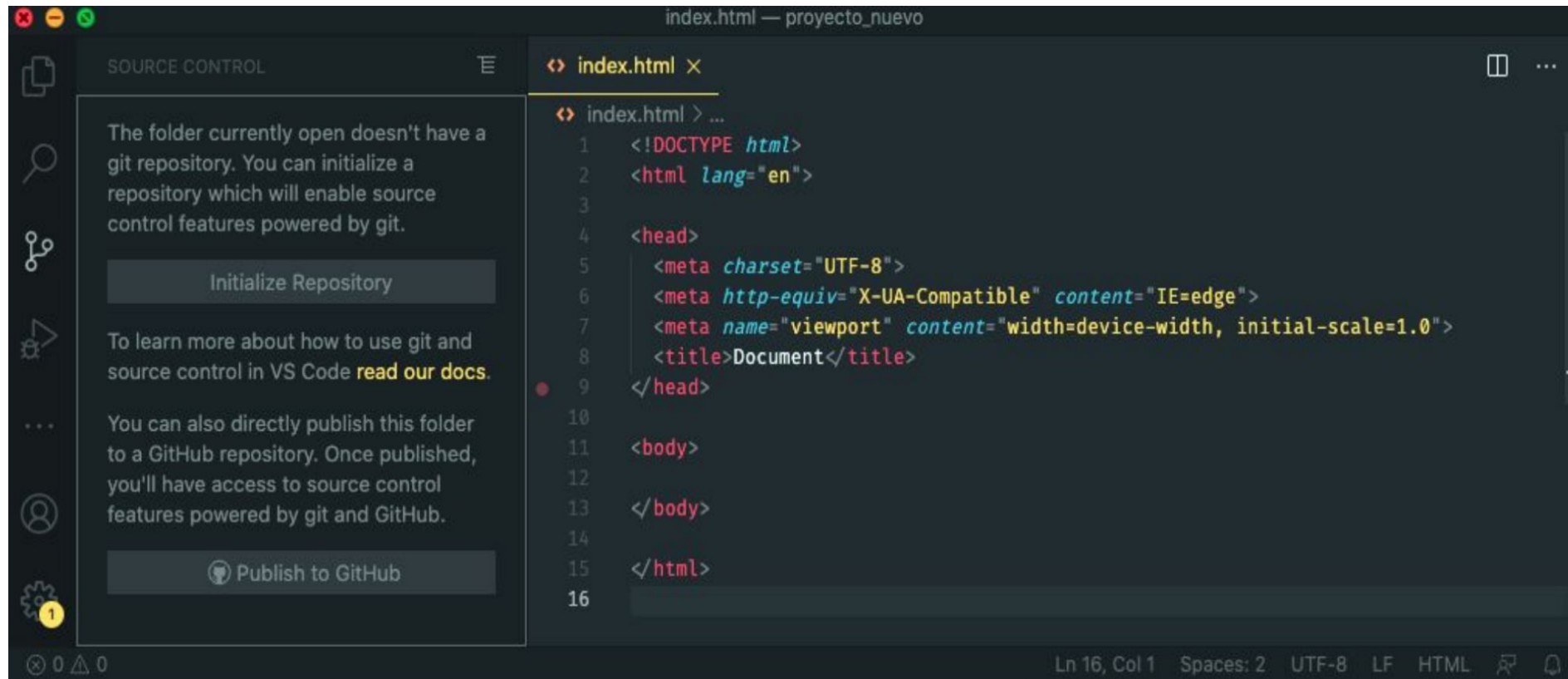
Git - Inicializando un repositorio

Un repositorio es un espacio en el cual podremos almacenar, organizar y mantener nuestro código.

Para crear un repositorio desde nuestro editor, debemos ir a la pestaña de Source Control y darle click en Initialize Repository (inicializar repositorio).

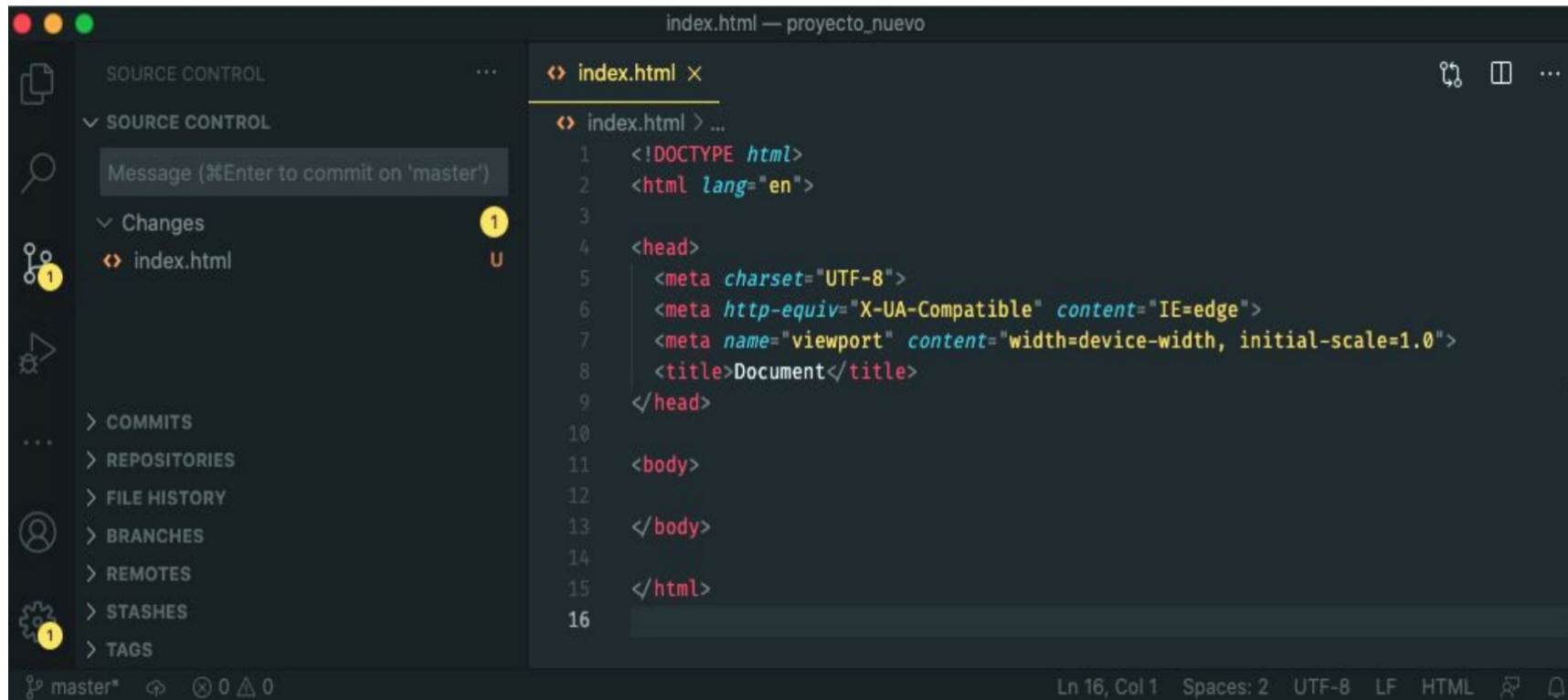
Podemos hacerlo también desde la terminal escribiendo “git init”

Git - Inicializando un repositorio



Git - Inicializando un repositorio

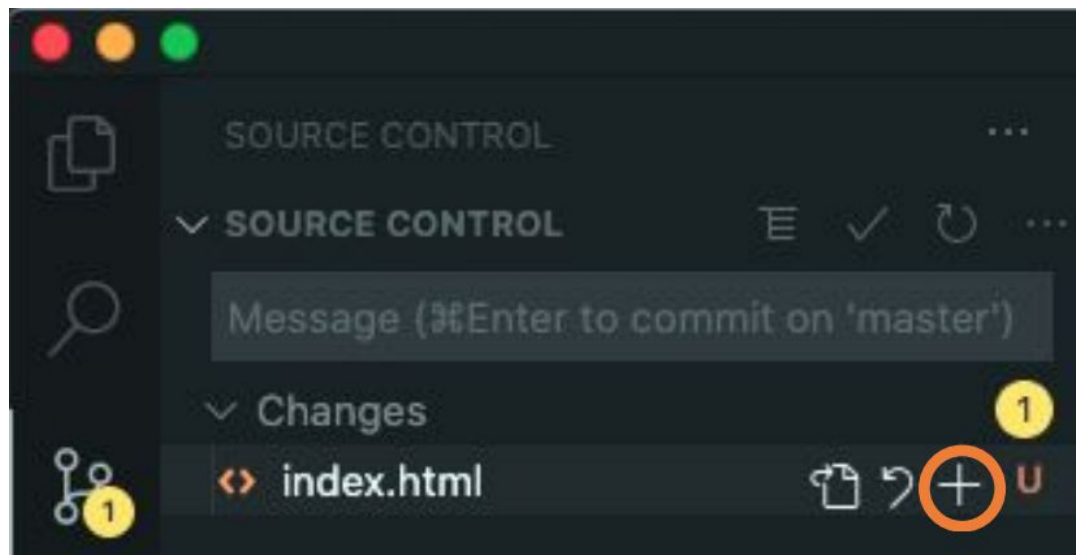
Una vez inicializado, nos encontraremos con una vista similar a la de la foto:



Git - Agregando cambios

Enfocándonos en la barra de la izquierda, podemos agregar archivos al siguiente bloque de cambios (o **área de “staging”**) clickeando en el “+” al lado de cada archivo.

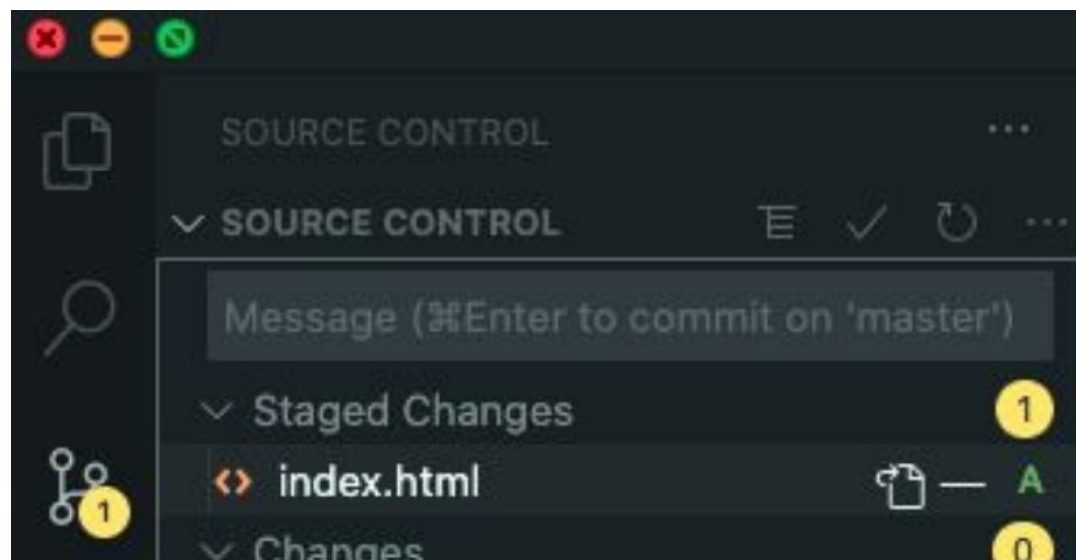
Desde la terminal: **git add index.html** (con tantos archivos como sea necesario)



Git - Agregando cambios

Una vez agregados, todos los archivos que estén listos para ser partes de la próxima versión estarán juntos en una lista.

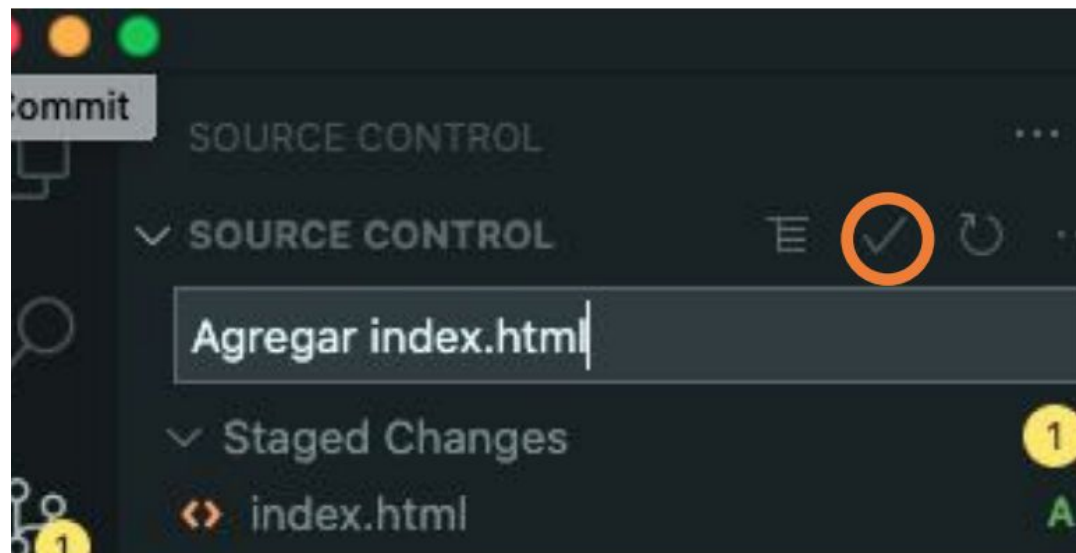
Este es un buen momento para revisar que todo lo agregado está ok.



Git - Agregando cambios

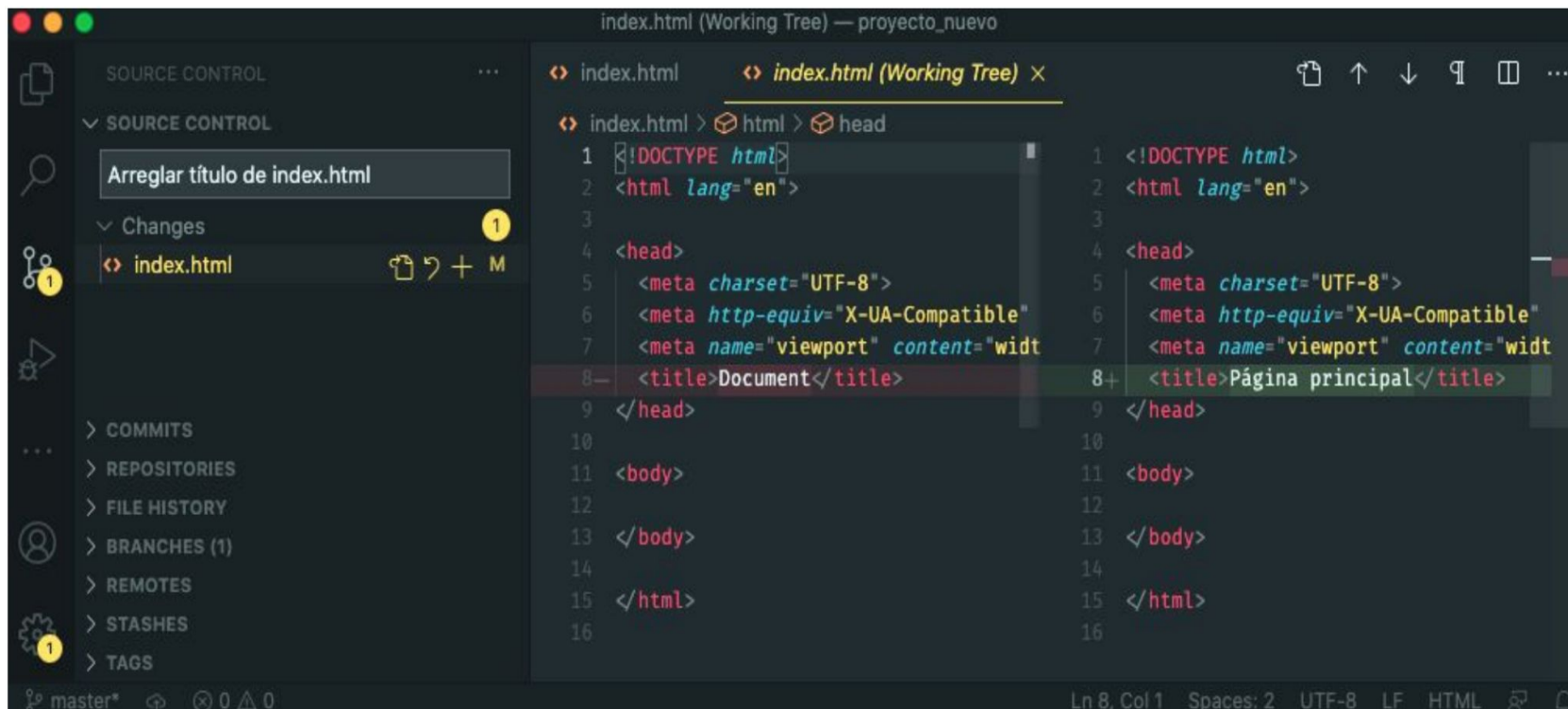
Luego de revisar los cambios debemos **asociar un mensaje** a nuestros cambios. Esto lo podemos hacer desde el VS Code escribiendo en la cajita que aparece sobre la lista.

Luego de escribir el mensaje, debemos presionar en el botón de 
Desde la terminal: **git commit -m "Agregar index.html"** (único paso)



Git - Agregando cambios

Cada vez que estemos trabajando en nuevos cambios podremos ver “la diferencia” entre lo que había antes y lo que estamos a punto de agregar:



The screenshot shows the Visual Studio Code interface with the Source Control panel on the left and the diff view in the center. The Source Control panel shows a commit message "Arreglar título de index.html" and a list of changes including "index.html". The diff view shows the difference between the current index.html and the index.html (Working Tree). The diff highlights the change in the title tag from "Document" to "Página principal".

```
index.html (Working Tree) — proyecto_nuevo
index.html > html > head
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta http-equiv="X-UA-Compatible">
7   <meta name="viewport" content="width=device-width, initial-scale=1">
8-  <title>Document</title>
8+  <title>Página principal</title>
9 </head>
10
11 <body>
12
13 </body>
14
15 </html>
16
```

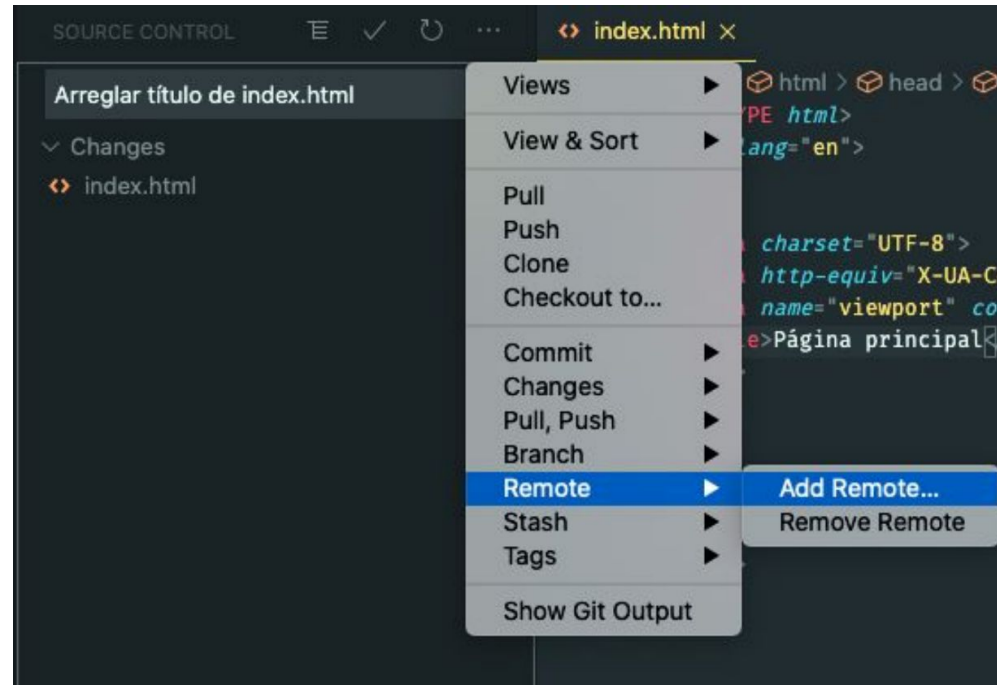
Git - ¿Remotos?

Uno de los grandes ventajas de Git es la facilidad de **trabajar en equipo de manera colaborativa** - no sólo con otras personas pero con “nosotros mismos” en distintos dispositivos.

Para trabajar de manera descentralizada **necesitamos crear un repositorio remoto**, lo cual podremos hacer en GitHub de la misma manera que lo hicimos con la entrega.

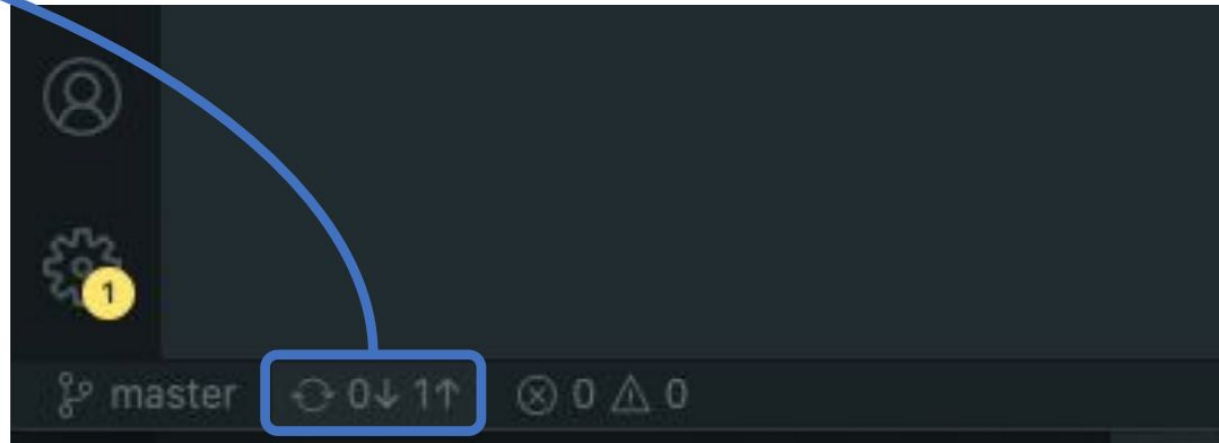
Git - ¿Remotos?

Una vez creado, podemos agregarlo a nuestro repositorio local dando click en el menú punteado, yendo a **Remote -> Add Remote...** (también podemos seguir los pasos que muestra GitHub en repositorios vacíos)



Git - Publicar en un remoto

Lo último que nos queda es publicar los cambios **desde nuestro repositorio local a nuestro repositorio remoto**, lo que podremos hacer desde VS Code clickeando en el siguiente botón (nota: puede aparecer una nube si es la primera vez)



Vamos a decir que cada **rama** o **branch** en Git es como una **línea de tiempo paralela**.

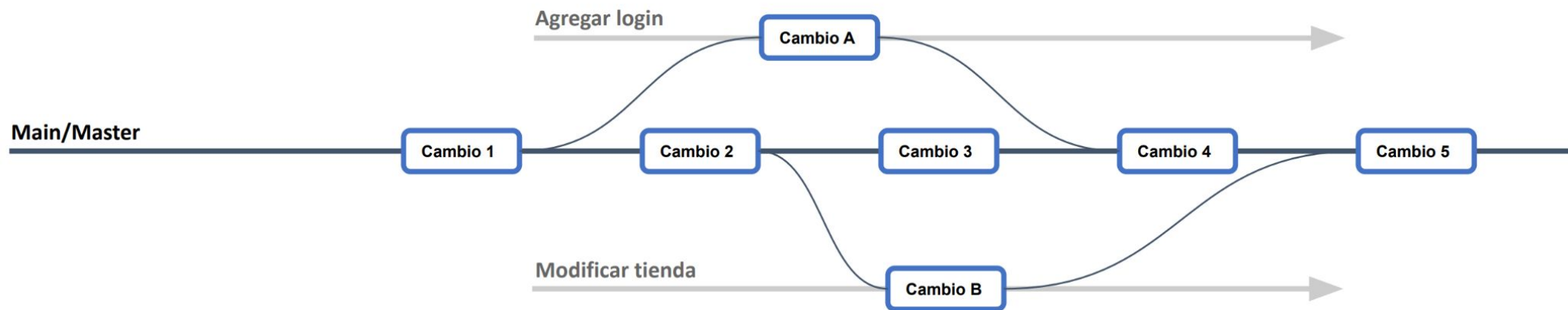
Una rama se puede crear manualmente a partir de un commit, pero luego de crearse estará separada completamente hasta que decidamos unirla a otra rama.

Si bien cada equipo elige su modalidad de trabajo, lo más normal es tener una rama base (llamada **main** o **master**) y realizar todos los cambios en ramas paralelas que, luego de ser validadas, se “unirán” a master.

Tengan en cuenta que esto es una simplificación y que las ramas en realidad nunca se unen a otras, sino que los cambios presentes en una rama se pueden llevar a otra, normalmente con un flujo llamado **PR (Pull Request)**.

Git - Branches

El siguiente diagrama representa un flujo normal en un proyecto:



Git - git clone

Este comando involucra la creación de un repositorio totalmente nuevo en GitHub, clonarlo en nuestra computadora, trabajar en nuestro proyecto y enviarlo de regreso.

Utilizando este método no es necesario ni inicializar el repositorio localmente, ni agregar un remote, ya viene configurado el mismo, pronto para usarlo.



```
$ git clone https://github.com/gustavgueez/senpai-fsd8.git
```

LINKS

LINKS

- **Git y github**
 - <https://github.com/>
 - <https://git-scm.com/>
 - <https://git-scm.com/book/es/v2/Inicio---Sobre-el-Control-de-Versiones-Con-figurando-Git-por-primera-vez>
 - <https://www.freecodecamp.org/espanol/news/guia-para-principiantes-untitled/>
- **Extensión VS code**
 - <https://marketplace.visualstudio.com/items?itemName=mhutchie.git-graph>



[gustavgueez](#)



[gustavgueez](#)



[gustavgueez](#)

GUSTAVO RODRIGUEZ

FULL STACK DEVELOPER
SOLCRE