

DESARROLLO DE APLICACIONES FULL STACK JAVASCRIPT TRAINEE



Git

Denis Pacheco



OTEC PLATAFORMA 5 CHILE

Origen

En el desarrollo de aplicaciones (y en general en cualquier trabajo sobre sistemas electrónicos) siempre ha existido la problema del trabajo en grupo.

Si hay varias personas trabajando sobre el mismo proyecto, se debe resolver dos principales factores :

- Compartir información
- Versión del proyecto



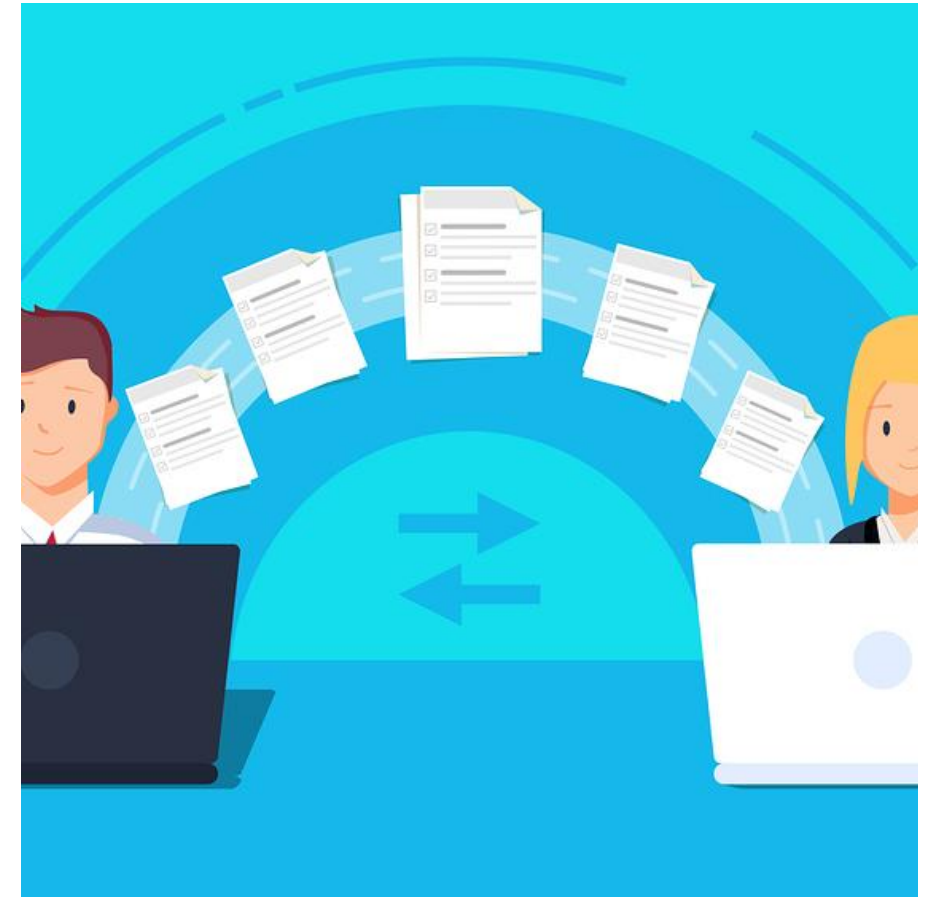
Compartir Información

Diariamente nos vemos obligados a compartir información a través de distintos medios:

Carpetas compartidas

Correo electrónico

Repositorios online (drive, Dropbox, etc..)



Versionado del proyecto

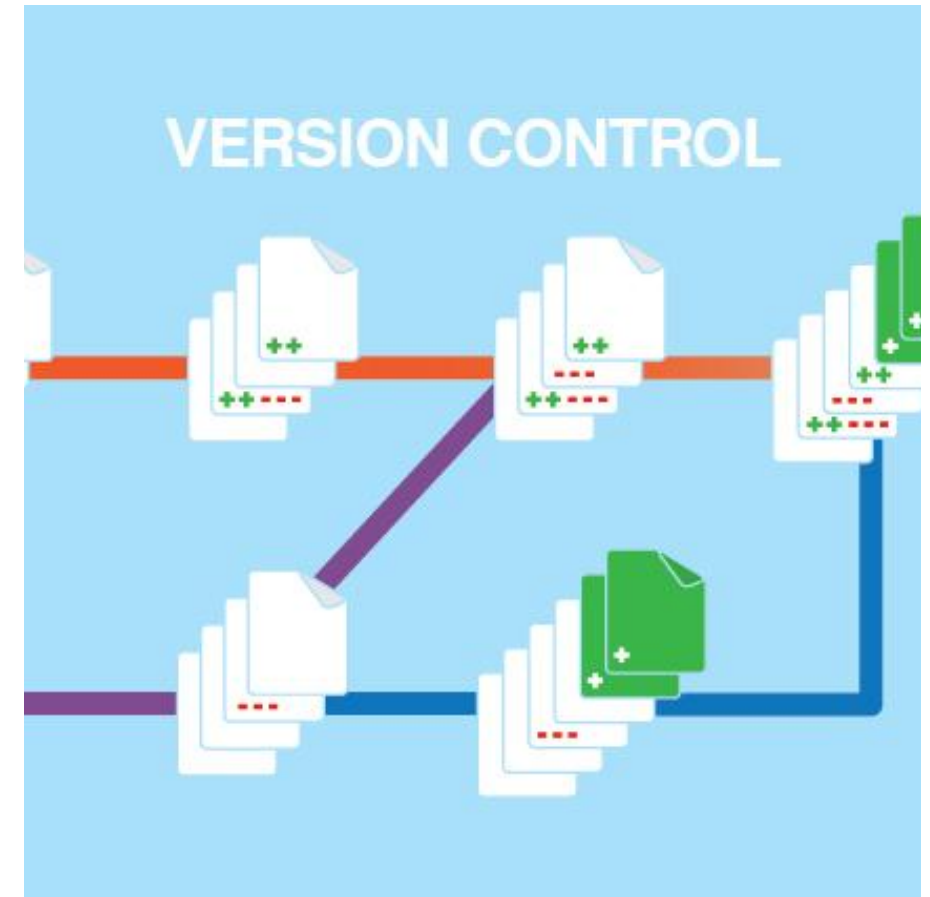
Las problemáticas a resolver aquí son:

Cuál de todos los archivos es el mas actualizado?

Cuál es el que mejor funciona?

Quien es el autor del archivo?

Quién fue el último en modificar un archivo?



Git

Git fue creado precisamente para resolver estos dos problemas de forma simultánea.

Su origen radica en el auge del software de código abierto y la necesidad de que una comunidad se encargue del desarrollo de un sistema completo



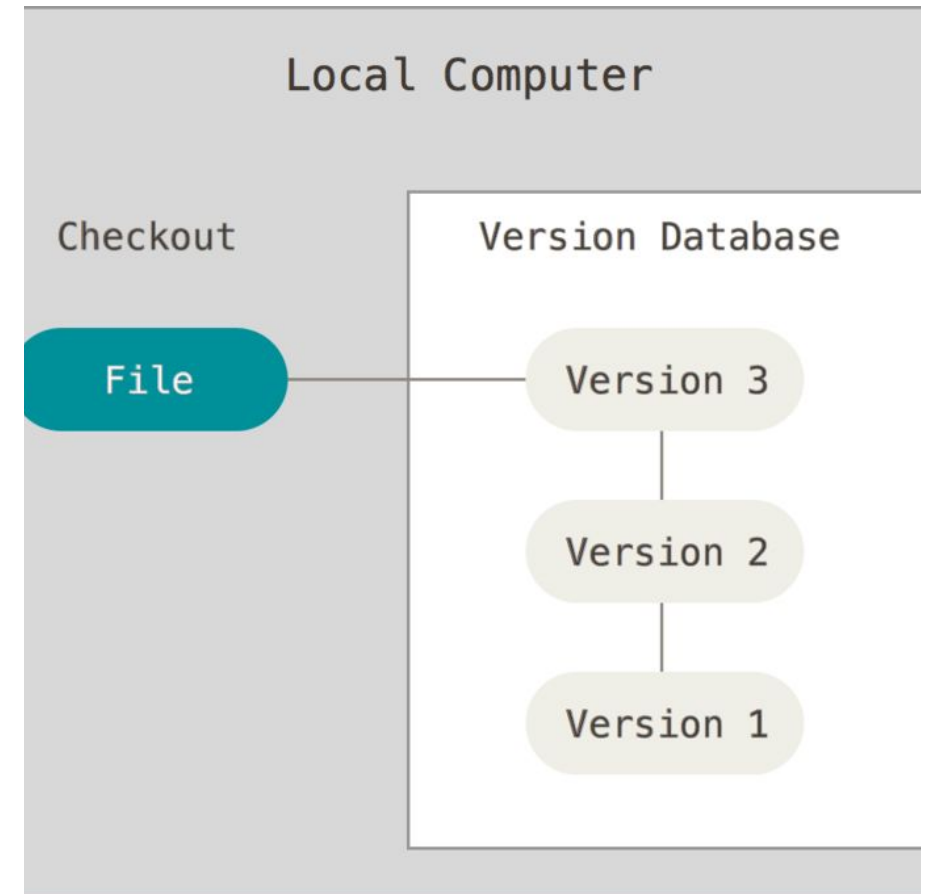
Cómo funciona?

Básicamente, git guarda los cambios que se van realizando en cada archivo del proyecto.

Lleva un historial de cambios (fecha, versión y autor)

Lo guarda en un “repositorio centralizado”

Esto permite la coordinación entre los distintos desarrolladores del proyecto.



Github

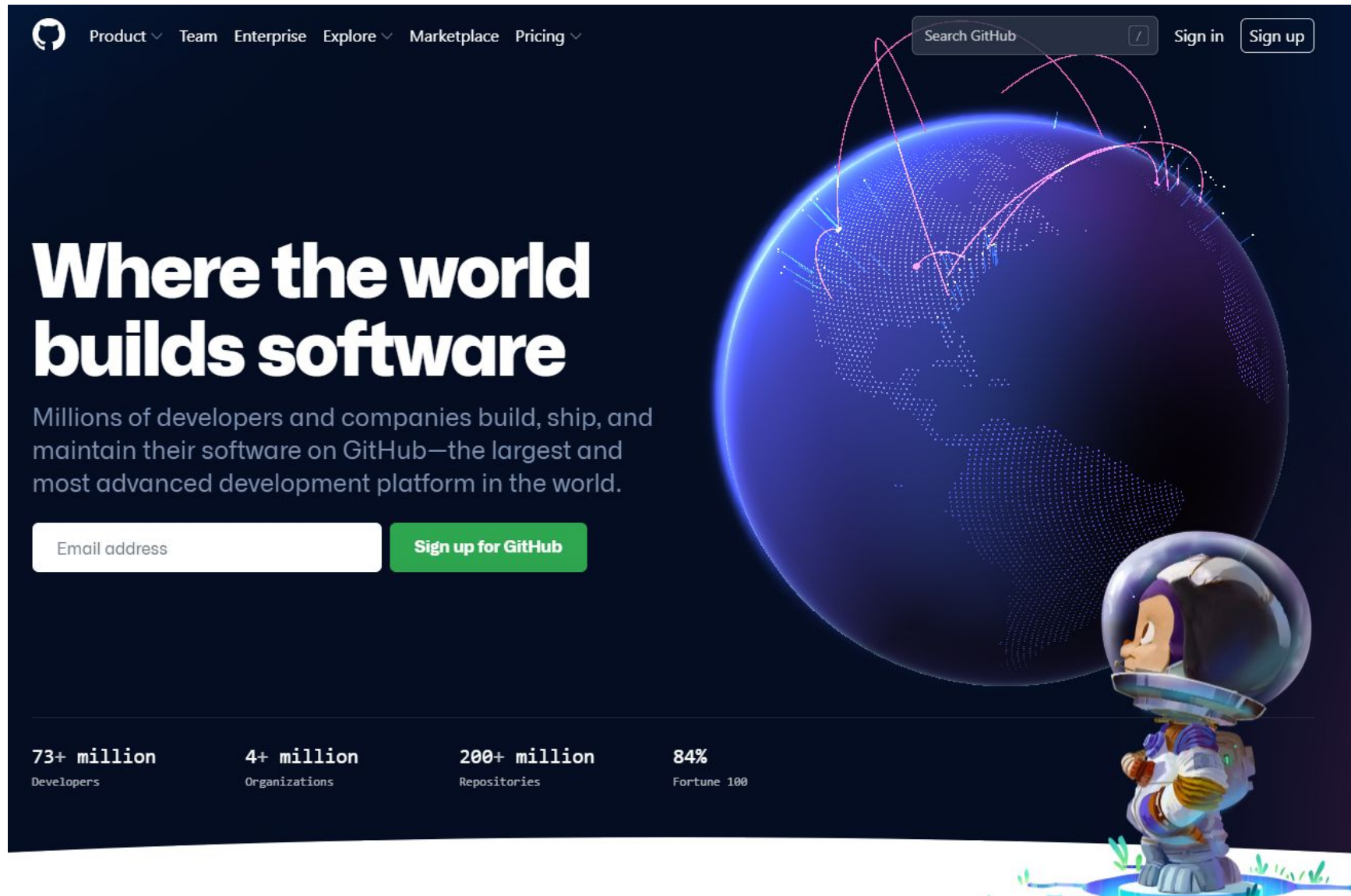
Github es una “evolución” o “implementación” de Git

Es una plataforma de trabajo colaborativo para almacenar proyectos en la nube, utilizando el control de versiones de git.

Fue creado en 2008 y desde el 2018 es administrada por Microsoft.

No es la única plataforma git, pero si es la mas importante.



The image is a screenshot of the GitHub homepage. At the top, there is a navigation bar with the GitHub logo, links for Product, Team, Enterprise, Explore, Marketplace, and Pricing, a search bar labeled 'Search GitHub', and buttons for 'Sign in' and 'Sign up'. The main section features a large, stylized blue globe with glowing red orbital lines. To the left of the globe, the text 'Where the world builds software' is displayed in a large, bold, white font. Below this, a smaller line of text states: 'Millions of developers and companies build, ship, and maintain their software on GitHub—the largest and most advanced development platform in the world.' Underneath this text is a white input field labeled 'Email address' and a green button labeled 'Sign up for GitHub'. At the bottom of the page, there are four statistics: '73+ million Developers', '4+ million Organizations', '200+ million Repositories', and '84% Fortune 100'. On the right side of the page, there is a small, stylized illustration of a person in a space suit looking up at the globe. To the right of the main image, there is a blue plus sign and a blue dot.

Product Team Enterprise Explore Marketplace Pricing

Search GitHub Sign in Sign up

Where the world builds software

Millions of developers and companies build, ship, and maintain their software on GitHub—the largest and most advanced development platform in the world.

Email address Sign up for GitHub

73+ million Developers
4+ million Organizations
200+ million Repositories
84% Fortune 100

Cómo funciona?

Para utilizar github sólo necesitan una cuenta que pueden crear en el mismo sitio web ;)

<https://github.com/>

Luego de esto, debemos configurar la conexión desde nuestro computador a través de visual studio code o cualquier software compatible con github



Algunas definiciones

Repositorio: Carpeta local o en la nube (github) donde guardaremos nuestros archivos

Revisión o versión: Es simplemente el estado de un archivo en cierto momento, luego de aplicadas modificaciones.

Etiqueta (tag): identificador de la versión (ej: v1.0, alpha, versión final, etc..)

Rama (branch): es una bifurcación, en la cual a través de la copia de una versión se sigue un camino paralelo de desarrollo.

Trunk (rama principal): es la línea de desarrollo original

Desplegar (checkout): es crear una copia local de nuestro proyecto a partir de un repositorio. En términos simples, sería “bajar los archivos” desde github para trabajar en un computador.

Confirmar (commit): confirmar los cambios realizados y guardarlos en el repositorio.

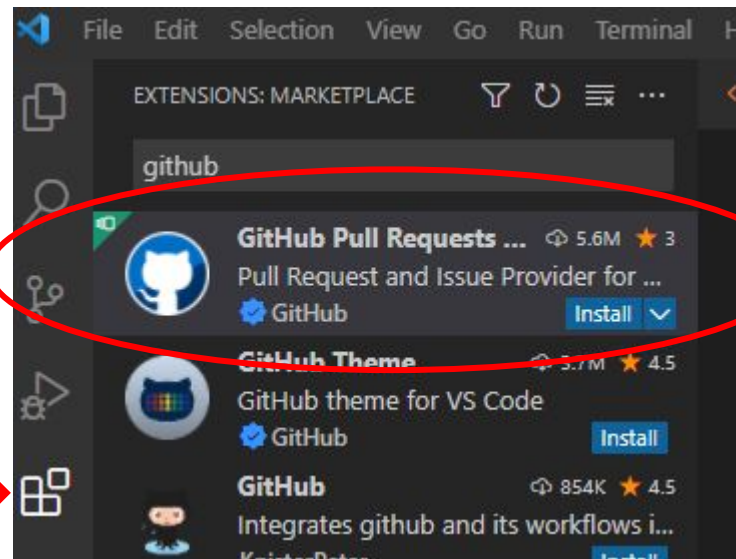
Conflicto: se produce cuando dos personas actualizan el mismo archivo al mismo tiempo.

Fusionar (merge): unir dos ramas distintas en un solo proyecto

Instalando Git en Vs Code

Lo primero que necesitamos para trabajar con github en Visual Studio Code es la extensión llamada “**Github Pull Request and Issues**”.

Esta la podemos encontrar en la sección “**extensions**”, escribiendo “GitHub” en el buscador



extensiones

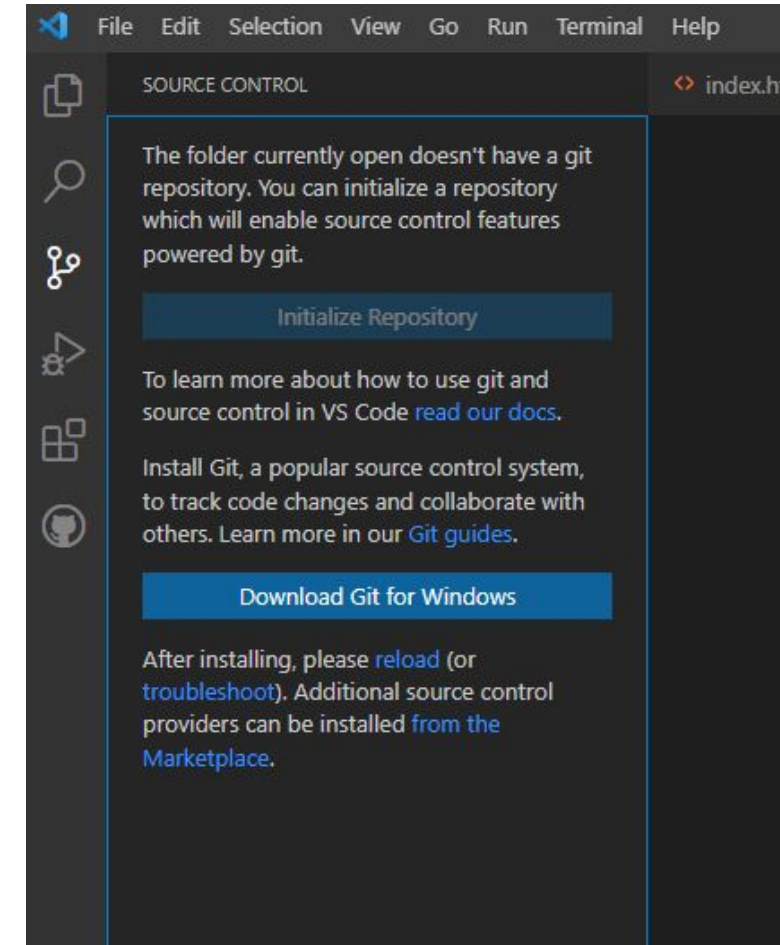


Instalando Git en Vs Code

Una vez instalado, aparecerán dos nuevos ítems en el menú lateral izquierdo.

El primero, nos pedirá instalar Git para poder utilizar la extensión

Hacemos click en “**Download Git for Windows**” y descargamos el software.

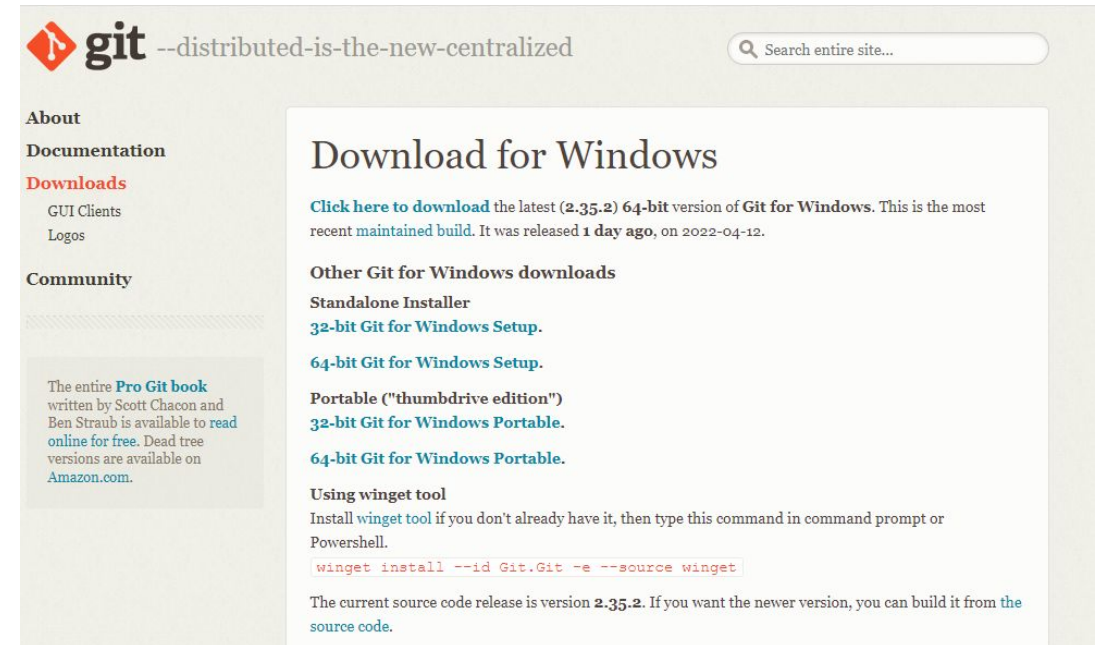


Instalando Git en Vs Code

Se nos abrirá una página web donde debemos elegir una versión de git.

Para la mayoría, debiera funcionar la versión “**64-bit standalone installer**”

La descargamos e instalamos

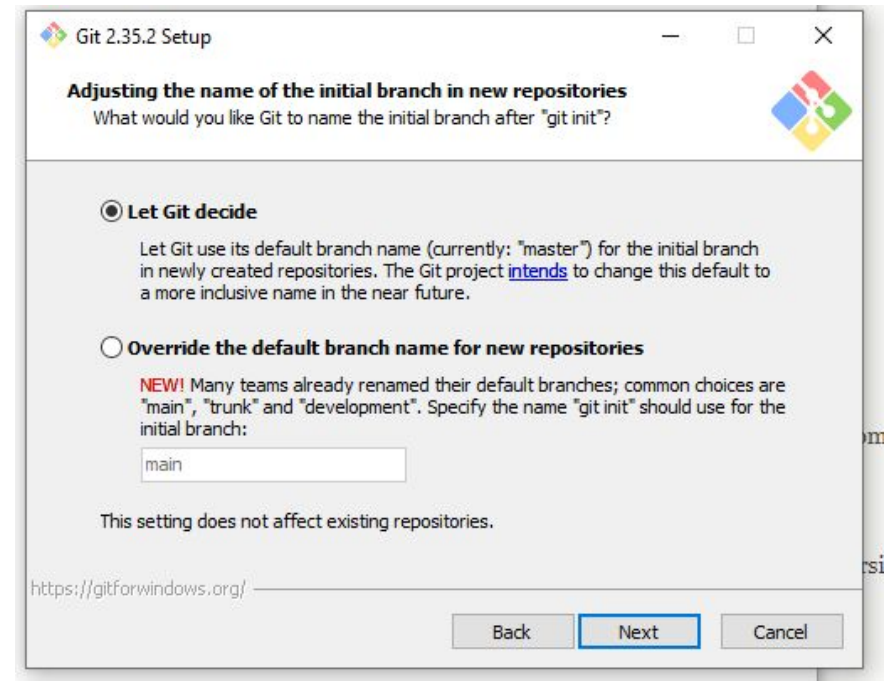


Instalando Git en Vs Code

En la instalación se nos preguntará por varias opciones.

En general dejamos todo como está y le damos “siguiente” o “next”

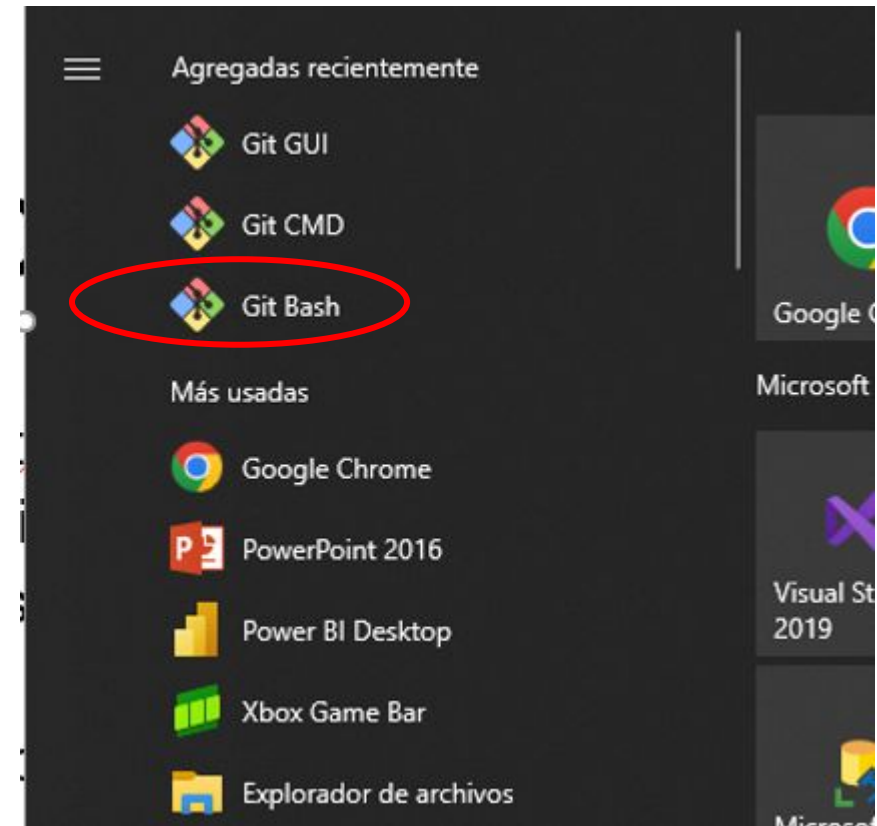
Al finalizar reinicien VS code para que detecte la instalación.



Configuración de Git

En git podemos hacer muchas cosas e incluso prescindir de github, pero para que el trabajo sea mas sencillo sólo vamos a configurar el usuario y email de git que son los únicos datos obligatorios para poder guardar nuestro archivos.

Para esto necesitamos abrir el programa Git Bash que se instaló en conjunto a Git



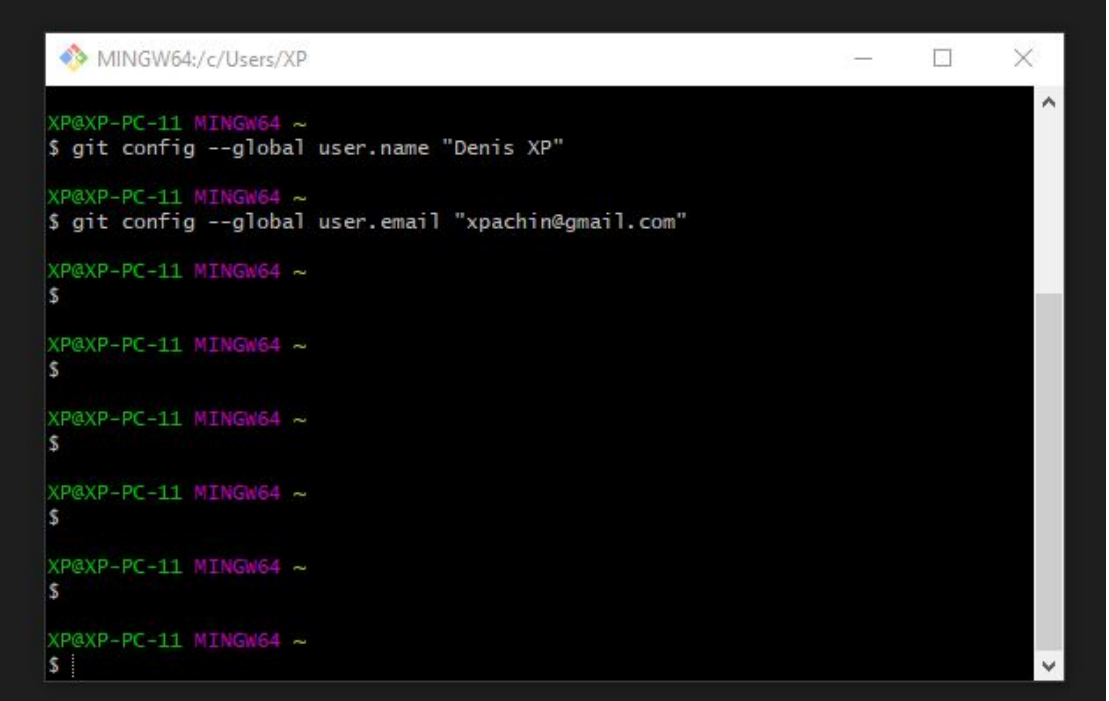
Configuración de Git

Se abrirá una consola en la cual escribiremos:

Git config --global user.name "su nombre"

Git config --global user.email "email de git"

Finalizamos y cerramos la ventana de consola
Git Bash



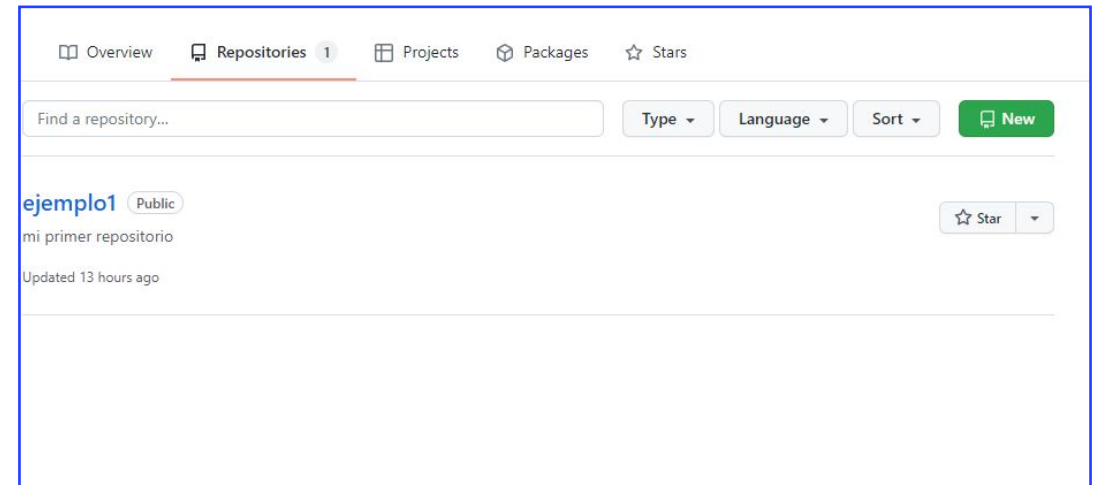
```
XP@XP-PC-11 MINGW64 ~  
$ git config --global user.name "Denis XP"  
  
XP@XP-PC-11 MINGW64 ~  
$ git config --global user.email "xpachin@gmail.com"  
  
XP@XP-PC-11 MINGW64 ~  
$  
  
XP@XP-PC-11 MINGW64 ~  
$  
  
XP@XP-PC-11 MINGW64 ~  
$  
  
XP@XP-PC-11 MINGW64 ~  
$  
  
XP@XP-PC-11 MINGW64 ~  
$  
  
XP@XP-PC-11 MINGW64 ~  
$  
  
XP@XP-PC-11 MINGW64 ~  
$
```


CheckOut

Hay varias formas para crear un repositorio, pero usaremos la mas fácil y común

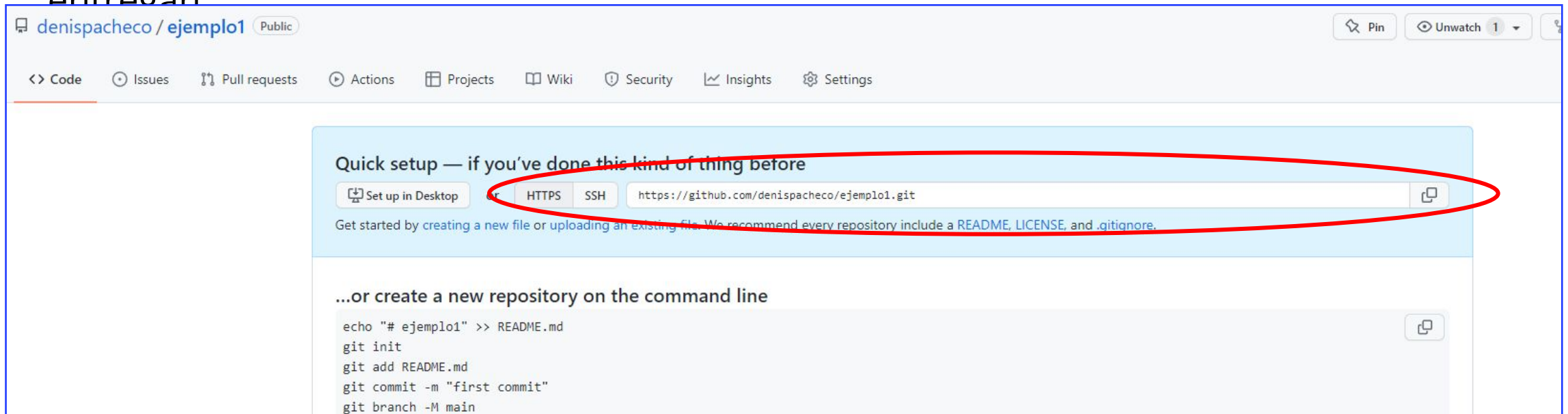
En github ya tenemos un repositorio creado ("ejemplo1"), el cual está vacío.

Vamos a crear una copia local ([checkout](#)) para trabajar en este y agregar nuestro archivos html!



CheckOut

Para esto entramos en nuestro repositorio y copiamos la dirección https que nos entregan:



The screenshot shows the GitHub interface for a repository named 'ejemplo1' by user 'denispacheco'. The 'Code' tab is selected. In the 'Quick setup' section, the 'HTTPS' option is chosen, and the URL 'https://github.com/denispacheco/ejemplo1.git' is displayed in a text box, which is circled in red. Below this, there is a section for creating a new repository on the command line with a list of commands.

denispacheco / ejemplo1 Public

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Quick setup — if you've done this kind of thing before

Set up in Desktop or **HTTPS** SSH `https://github.com/denispacheco/ejemplo1.git`

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

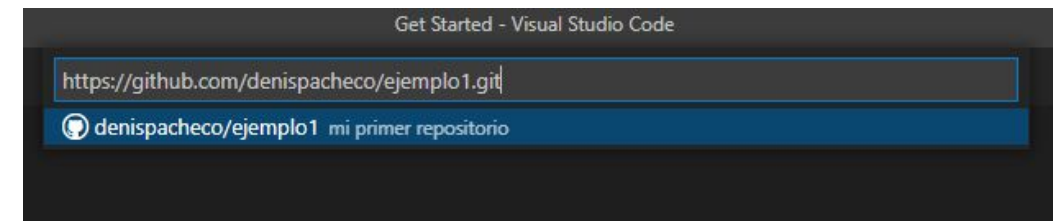
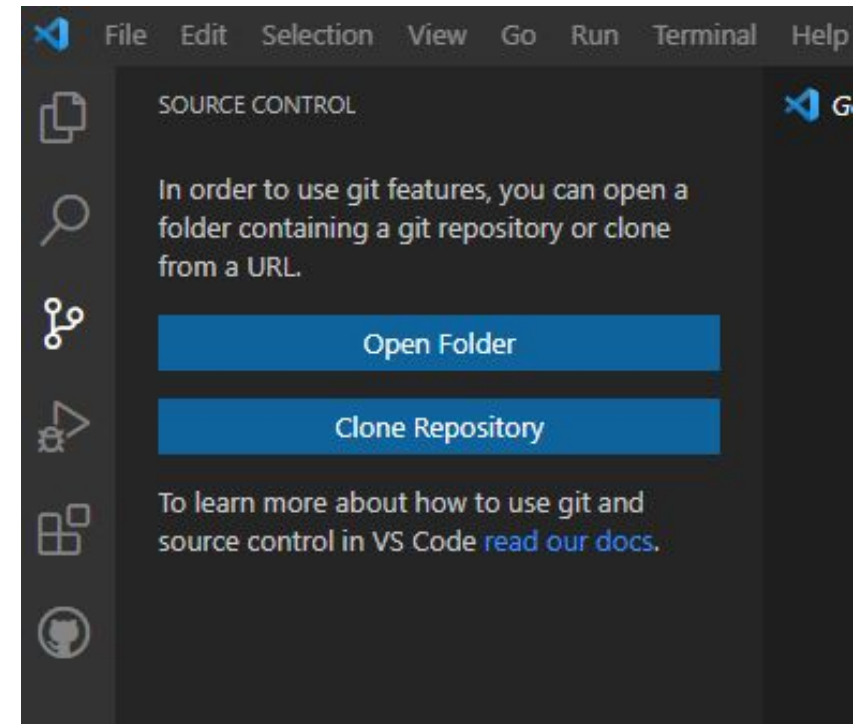
```
echo "# ejemplo1" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
```

Checkout

Nos Volvemos a Visual Studio Code y cerramos todas las carpetas o archivos que tengamos abiertos (File->Close Folder).

Ahora en el menú “**Source Control**” tendremos la opción “**Clone repository**”

La seleccionamos y pegamos la dirección copiada anteriormente cuando nos la pida o simplemente le damos a “**Clone from Github**” y las seleccionamos.

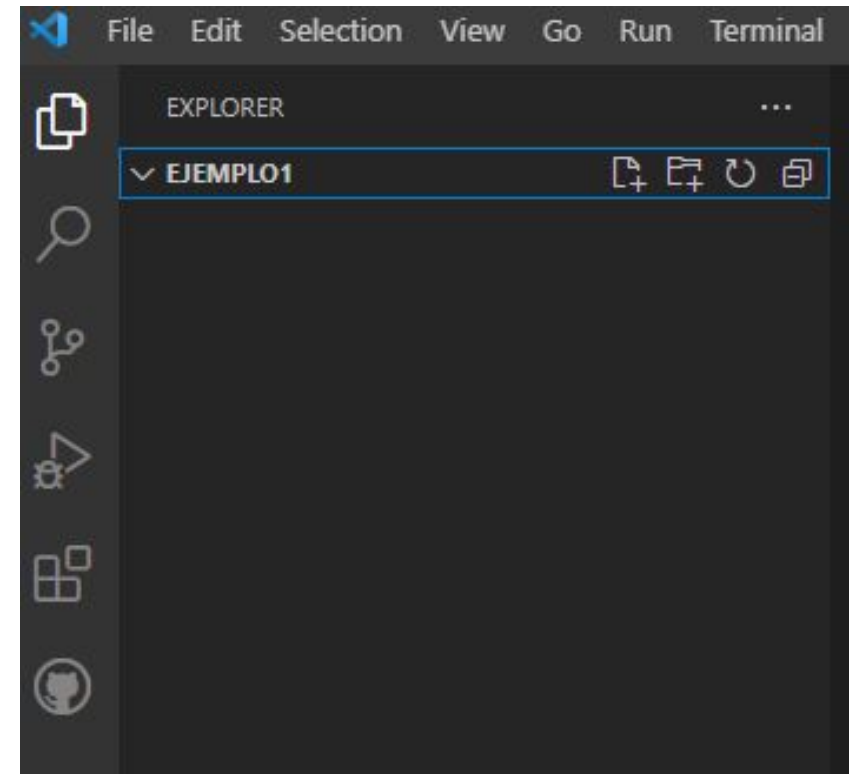


Checkout

Al hacerlo, debemos seleccionar la carpeta donde guardaremos nuestro proyecto.

Ojo que es una carpeta contenedora, ya que dentro de esta nos creará otra carpeta con el nombre del proyecto.

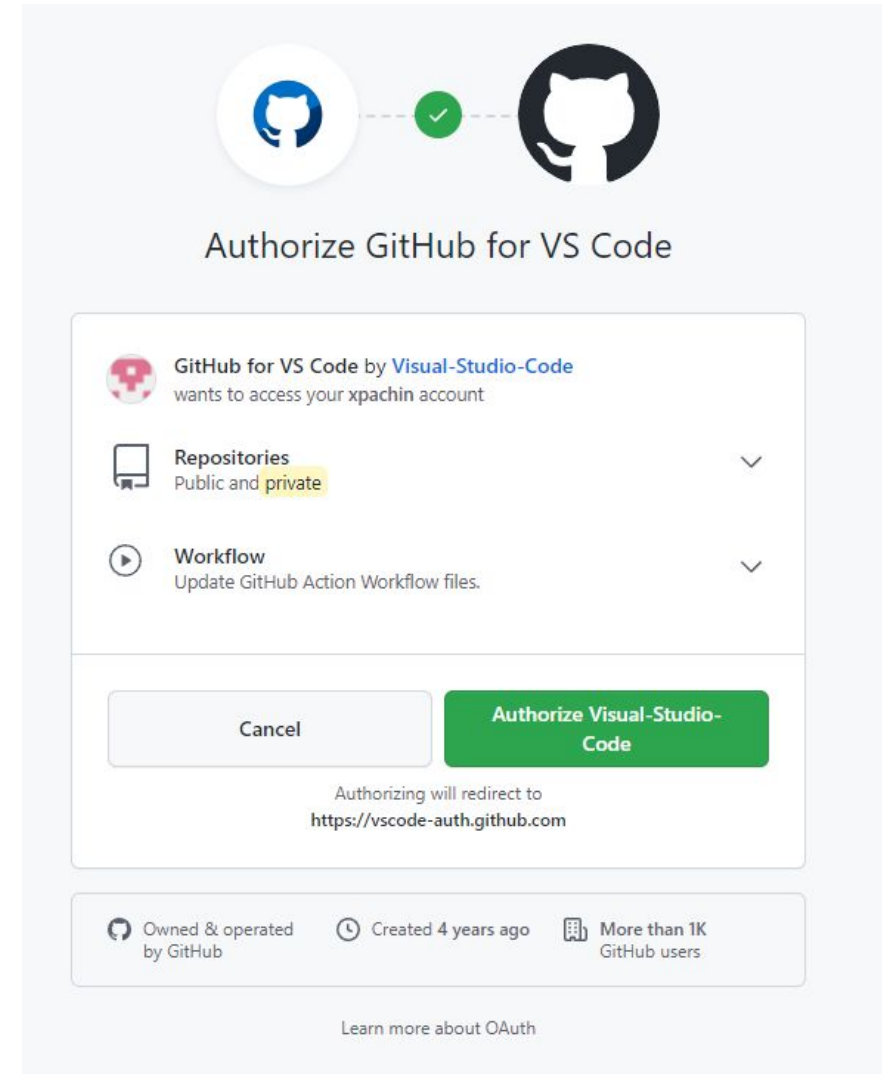
Luego nos preguntará si queremos abrir la carpeta y le decimos que si.



Checkout

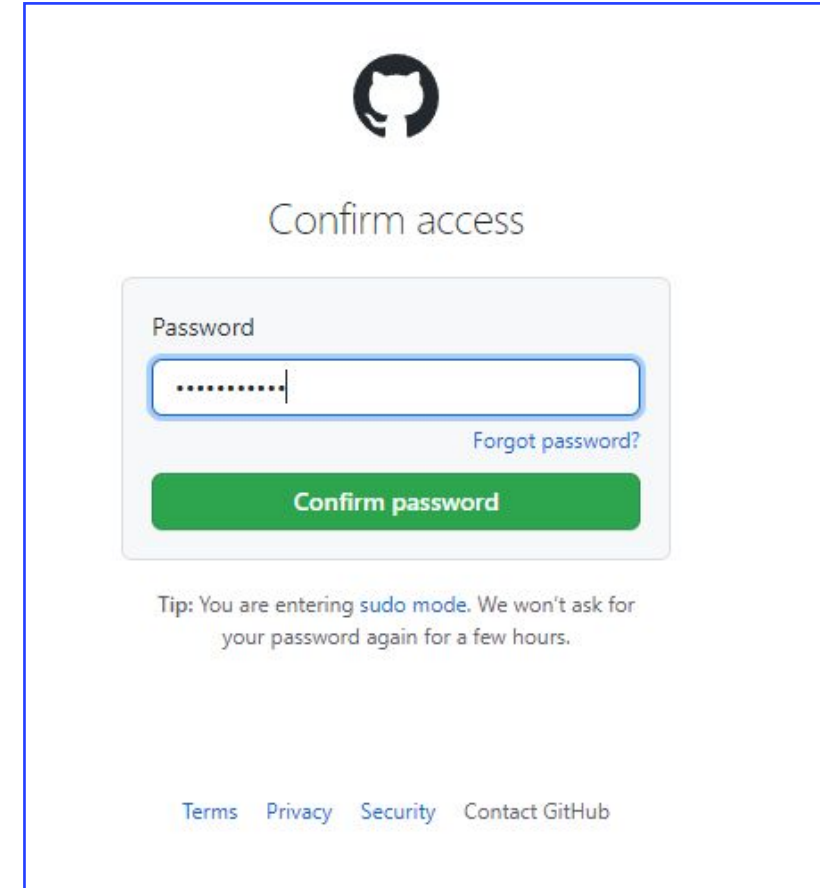
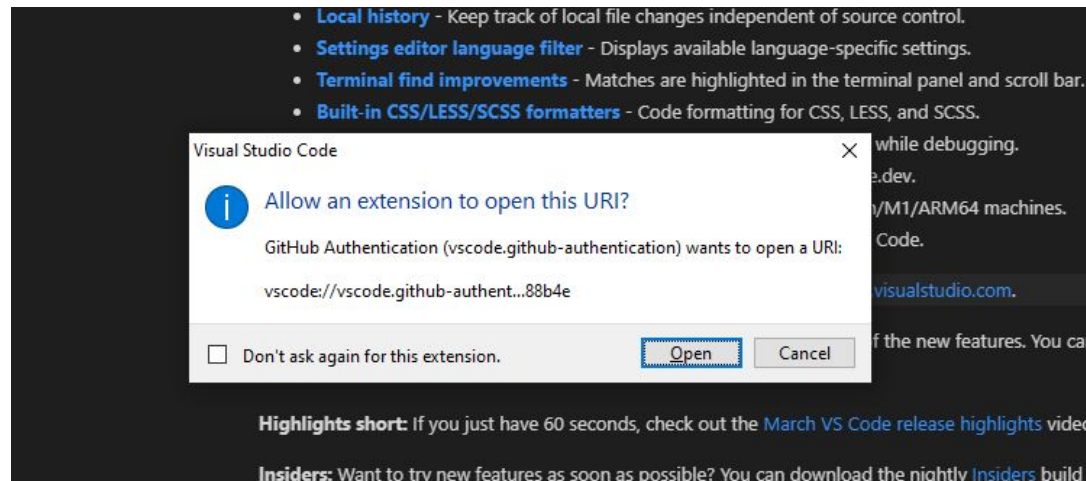
En algún momento puede que github les pida permiso por temas de seguridad. En este caso simplemente le dan al botón de autorización.

En este paso se recomienda reiniciar visual studio code para que los cambios tomen efecto



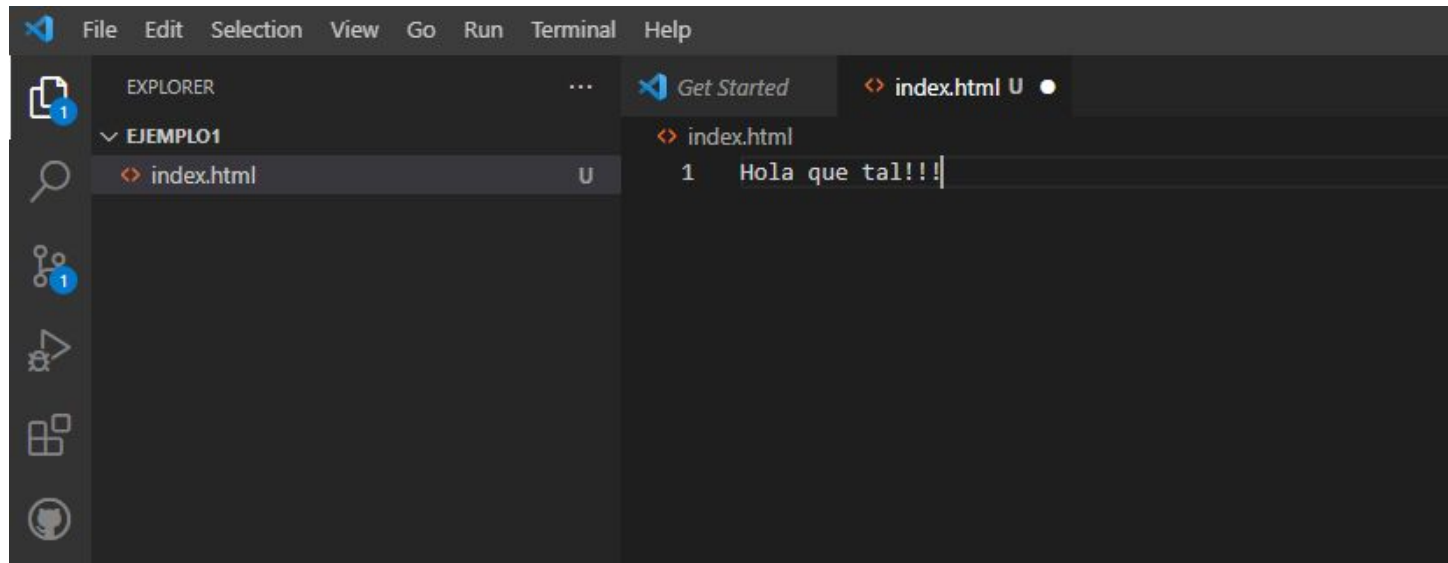
Checkout

También puede que les pida confirmar el acceso a github mediante su contraseña o aceptar que la extensión abra alguna url



Agregando código

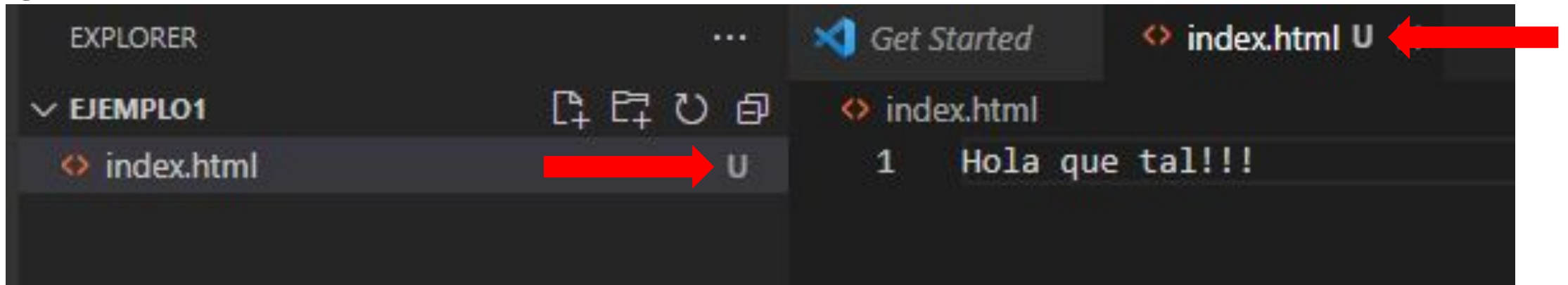
Ahora que tenemos nuestro repositorio preparado, agregaremos un archivo para trabajar con el siguiente paso, guardar el repositorio.



Agregando código

Notar que ahora nuestros archivos tendrán una letra en el lado derecho o en el título cuando estemos editándolo.

En este caso una “U”. Esta letra indicará el estado en nuestro repositorio, en este caso la U viene de “**unstaged**” o “**untracked**” lo que significa que no hemos preparado el cambio para guardarlo en git



Commit

Ahora que tenemos nuestro archivo, vamos a guardarlo en git!

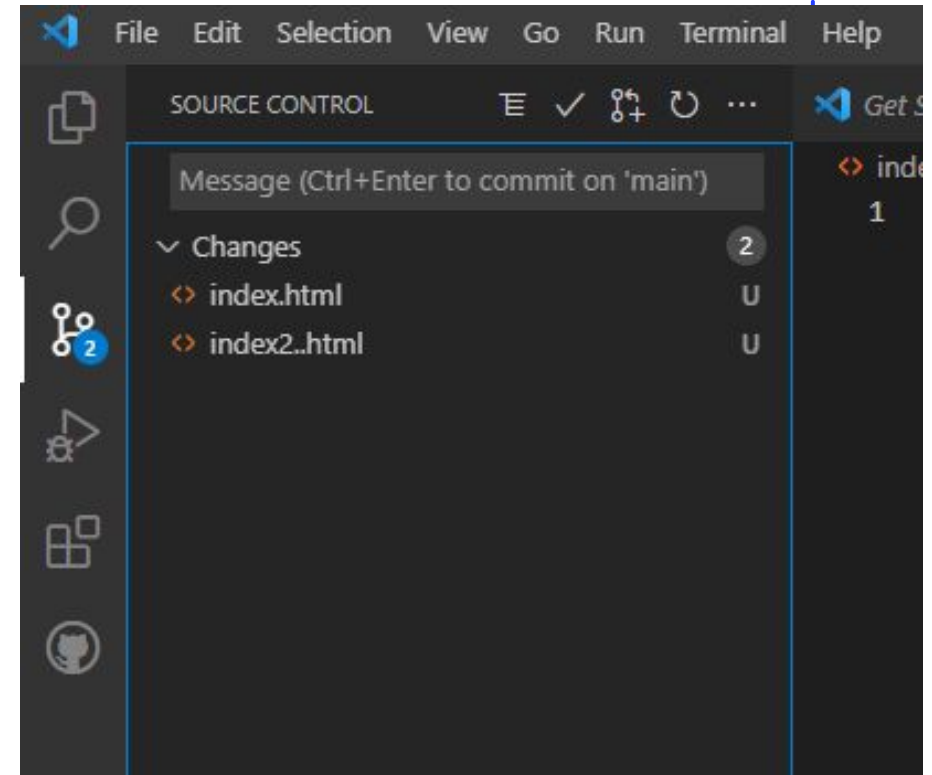
Cabe mencionar que NO es lo mismo guardar nuestro archivo (localmente) a hacer un **Commit** (guardarlo en git) o hacer un **Push** (github). Lo primero sólo afecta a nuestro archivo, lo segundo al repositorio y lo tercero al repositorio remoto.



Commit

Hay varias formas de hacer un **commit**:

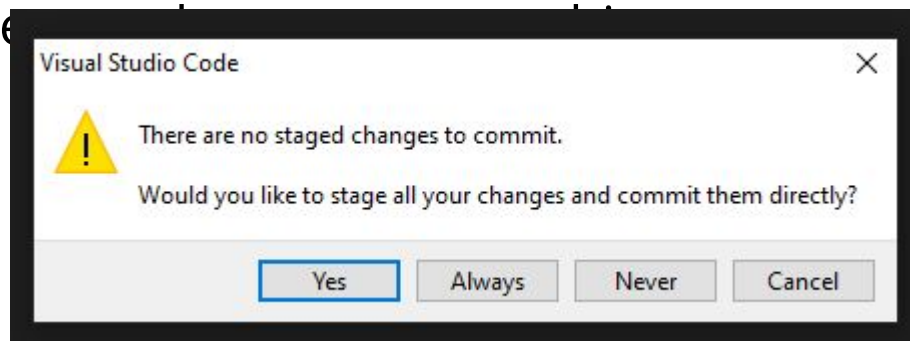
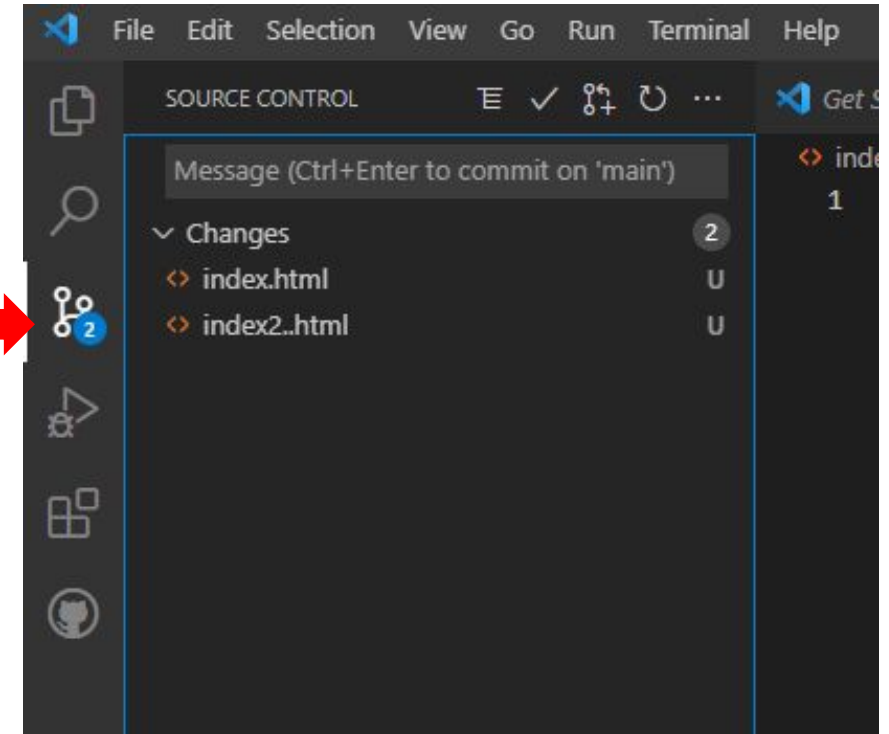
- Prepara los cambios, general o por archivo
- Hacer **commit** individualmente
- Hacer un **commit** general (todos los archivos modificados)
- Usaremos esta última fórmula para que el trabajo sea mas sencillo



Commit

Para ejecutar nuestro commit, nos vamos al menú "source control" y le damos click al botón **commit** ✓)

La primera vez nos advierte que no hay cambios preparados. Para facilitar las cosas, elegiremos "always" para haga el trabajo por nosotros y siempre



Commit

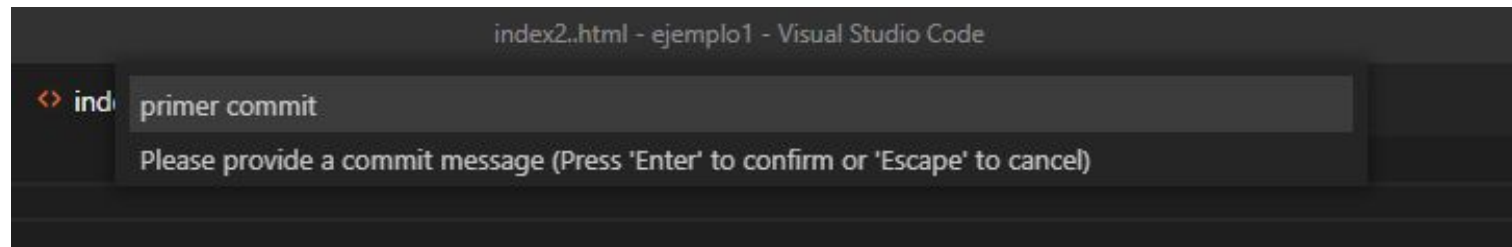
+

•

Todo **commit** genera una versión, por lo tanto debe tener la información de cuando y quién realizó los cambios (esto ya lo configuramos)

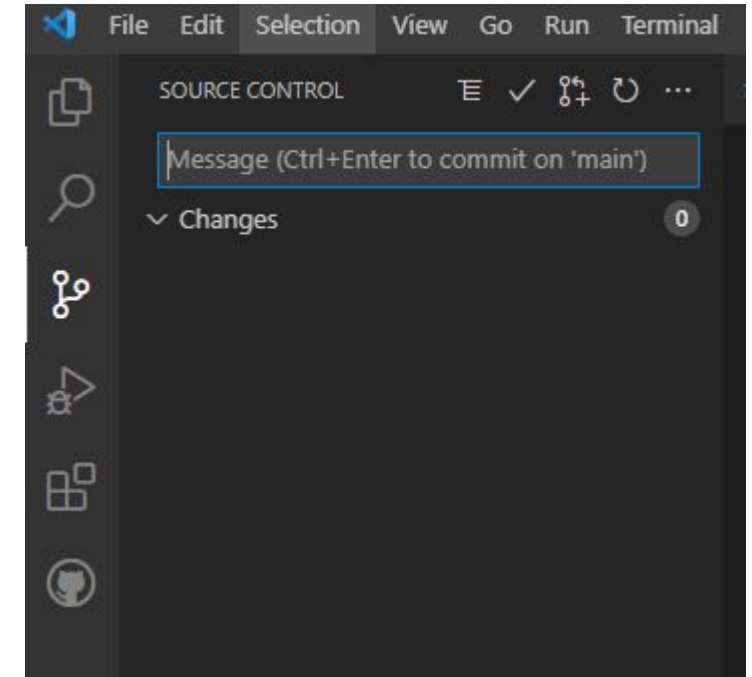
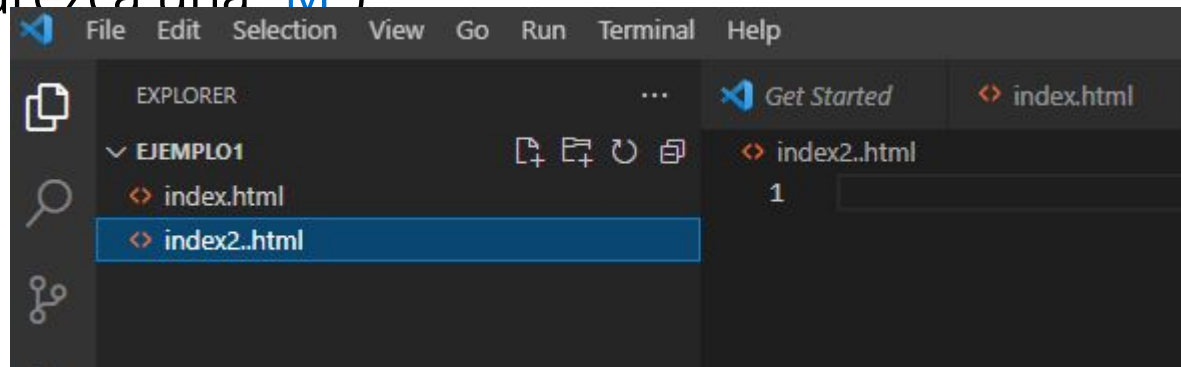
Pero además nos pide un comentario que describirá los cambios realizados en nuestro código. Esto es obligatorio siempre, y deben ingresar algún texto que describa el contenido de los cambios.

Para este ejemplo, podemos poner “**primer commit**”



Commit

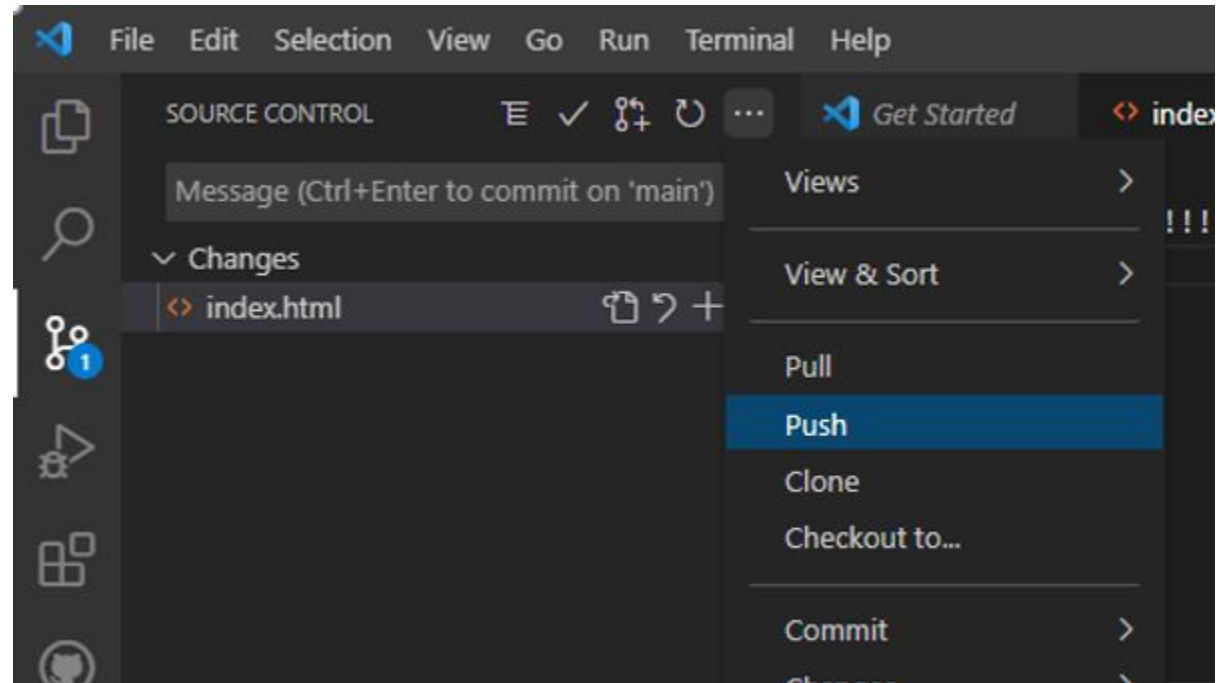
Si todo sale bien, en la sección “**source control**” desaparecerán todos los cambios y en la sección explorer (donde está nuestra carpeta), desaparecerá la letra “**U**” de nuestros archivos (hasta que los modifiquemos nuevamente y aparezca una “**M**”)



Push

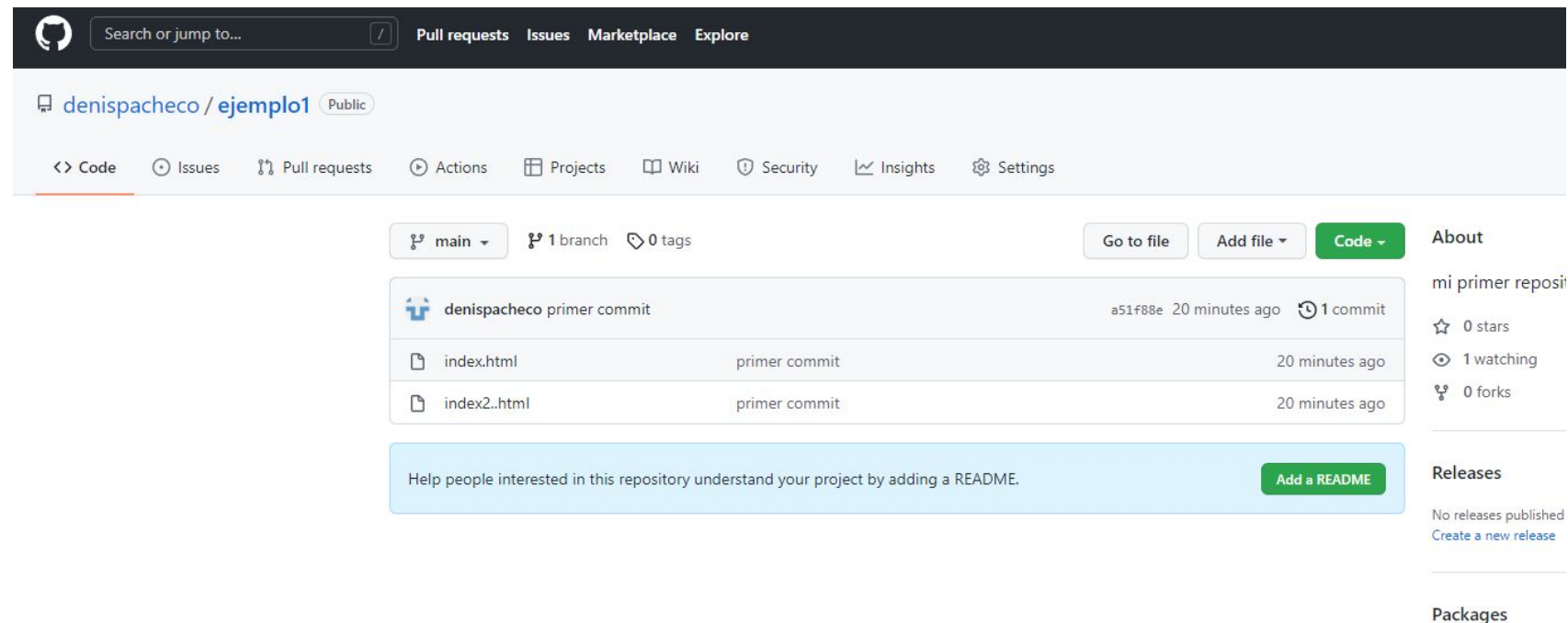
Sólo nos falta copiar nuestro repositorio a github, para esto nos vamos al menú de sourcecontrol y elegimos “**push**” desde el menú desplegable superior

Y si todo sale bien, ya tendremos nuestros archivos en la nube.



Github

Si revisamos nuestro repositorio, ya estarán los archivos que acabamos de guardar junto con el historial de cambios!



Github

De ahora en adelante el trabajo para guardar nuestros cambios en [Github](#) será siempre el mismo :

- Commit
- Push
- Y fin!

