Repositorio Git en Visual Studio Code con cuenta Github

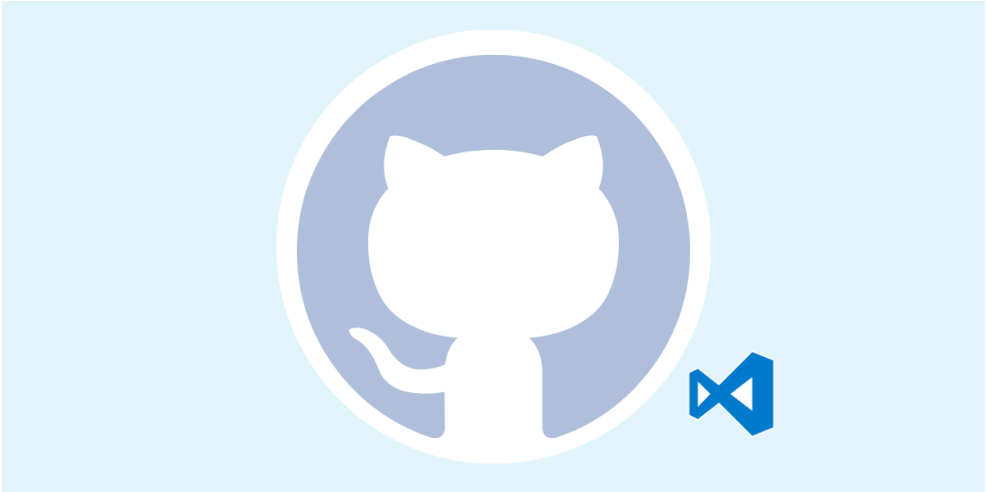


Tabla de contenidos

Como sabemos, Git y GitHub no son lo mismo, **Git** es nuestro **sistema local de gestión de versiones** y tiene la posibilidad de integración con otras plataformas como GitHub. Mediante la plataforma de ***social coding*** **GitHub** puedes **publicar repositorios de código** en remoto. De esta forma podremos trabajar con un **sistema de control de versiones** en la nube.

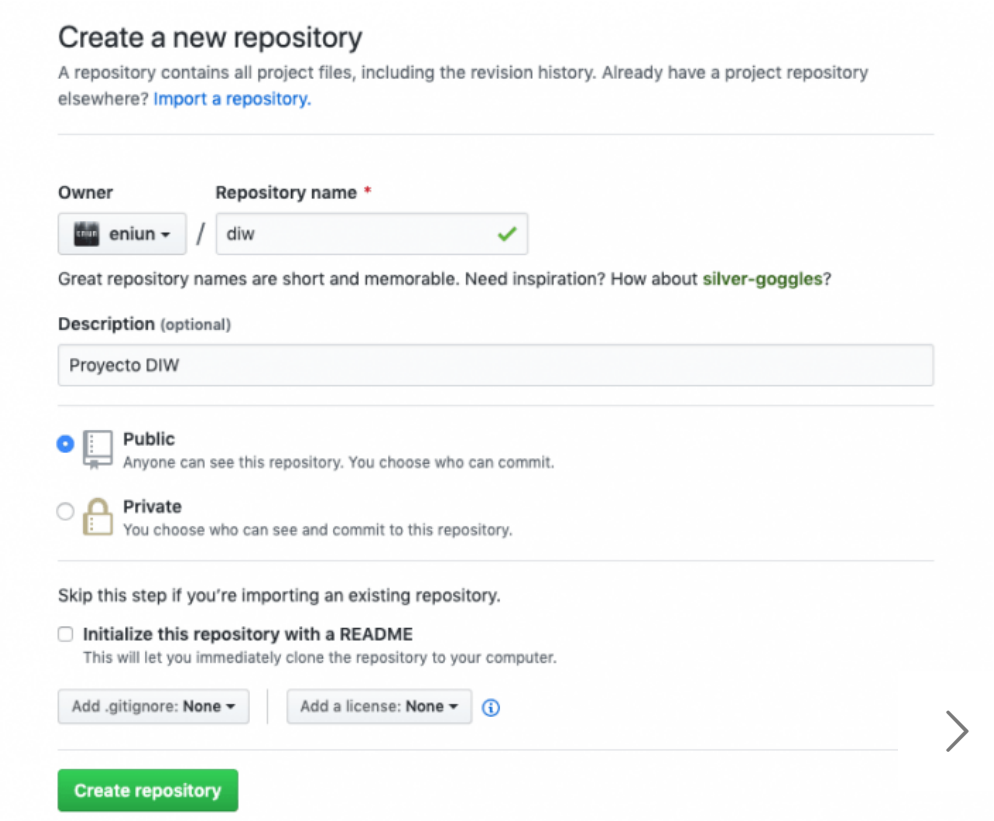
Este sistema te ofrece la posibilidad de **colaborar en otros proyectos y publicar los tu propios**. La plataforma es de **código abierto** por defecto, por lo que cualquier persona utilizar tu código y tú también puedes ver el código de otros proyectos. Este artículo es una **guía básica para publicar contenido usando GitHub y Visual Studio Code**.

13. Usar Github y Visual Studio Code para publicar contenido

13.1. Crear cuenta en Github

Accede a github.com y **crea una cuenta**. Selecciona el plan personal gratuito con repositorio público. No te olvides de terminar la verificación mediante correo electrónico.

Crea un proyecto en la opción "**Create repository**" o en «**Start a New Project**». Incluye el nombre de tu repositorio, selecciona «Public» y pincha el botón "Create repository". No cierres esta ventana porque vas a necesitar algunos de los datos que ahí se muestran para clonar el repositorio mediante URL. (Si por error cierras esta página, puedes volver a entrar mediante una url con el siguiente aspecto: **https://github.com/tunombredeusuario/nombredeturepositorio/** (en mi caso es la siguiente: https://github.com/eniun/diw/).



Unidades

. Planificación de faces gráficas

- UD2. HTML
- UD3. CSS básico

13.2. Clonar el repositorio con Visual Studio Code

Antes de comenzar a clonar el repositorio debemos instalar Visual Studio Code y Git.

13.2.1. Instalación

La **instalación de Visual Studio Code** se realiza desde la página oficial: code.visualstudio.com

La **instalación de Git** se realiza de diferente forma según el sistema operativo. Para instalar Git en **Linux** basta con teclear la siguiente línea en una ventana de terminal:

```
$ sudo apt-get install git
```

Para **instalar Git** en **otras plataformas**, puedes consultar los binarios disponibles en git-scm.com/downloads.

Una vez instalado Git, en cualquier sistema operativo, podemos comprobar que el programa se ha instalado correctamente comprobando el número de versión desde línea de comandos.

```
$ git --version
```

A continuación, debemos definir nuestro usuario e email de la siguiente manera:

```
$ git config --global user.name [nombre-usuario]
$ git config --global user.email [email]
```

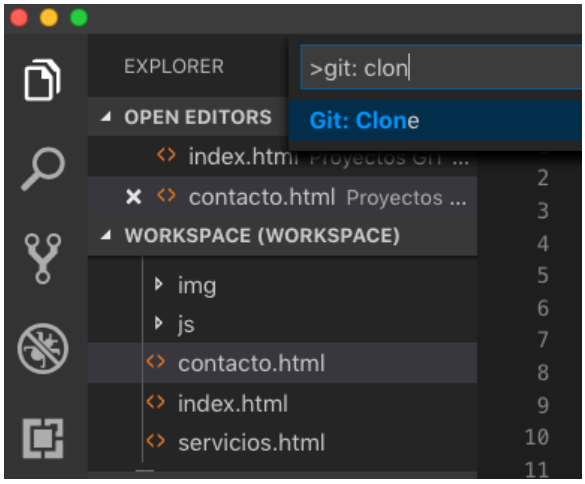

Mediante esta configuración, todo usuario que use esta sesión y esta máquina tendrá nombre eniun y el correo electrónico correo@correo.com. Este dato es el que aparecerá cuando se haga un commit.

> *Cuando hacemos un **commit**, estamos congelando el estado de todos los ficheros subdirectorios, y **guardando la instantánea en un histórico**. Debemos hacer un commit cuando queramos añadir un nuevo hito al listado de cambios del proyecto. En el commit debemos incluir un **mensaje sintetizado** pero descriptivo de los*

cambios que se han realizado. Por ejemplo: «Creación de la cabecera y el menú principal».

13.2.2. Clonación

Ahora vamos a **clonar el proyecto** creado en Github en nuestra máquina mediante Visual Studio Code. Para ello, nos vamos al **menú "view" seleccionamos «command palette»**. En ese punto buscamos "**git: Clone**". La herramienta nos pedirá la ruta y ahí es donde tenemos que pegar la URL del repositorio que hemos creado en Github. En mi caso: **https://github.com/eniun/diw.git**.

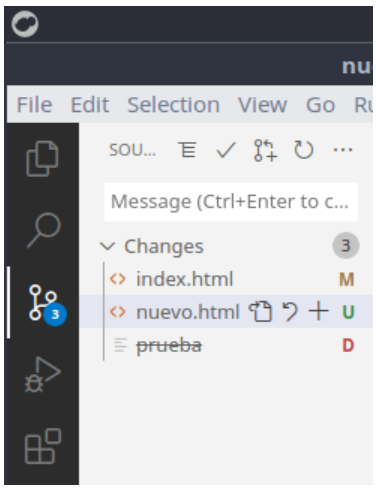


13.2.3. Autorización

Ahora debemos **autorizar nuestra cuenta de Github** en Visual Studio Code. Esto se puede hacer desde el icono de «**Accounts**» que aparece en la barra lateral izquierda de Visual Studio Code. No obstante, si nos saltamos este paso no hay ningún problema porque nos solicitará introducir los datos al hacer un commit.

13.2.4. Modificación de archivos

Ahora ya podemos empezar a **crear, eliminar o modificar archivos** del proyecto en nuestra computadora. Desde la pestaña de **Source Control** podemos ver todos los cambios que hemos hecho. En el ejemplo siguiente se puede ver que se han realizado tres cambios. Se ha creado el documento nuevo.html (**U, untracked**, sin seguimiento), se ha modificado el documento index.html (**M, modified**, modificado) y se ha eliminado el archivo prueba (**D, deleted**, eliminado).



Estados de los ficheros

- Untracked (Sin seguimiento)
- Tracked (Bajo seguimiento)
- Staged (Preparado para confirmación)
- Modified (Modificado)
- Deleted (Elminiado)

Áreas de trabajo

El modo de trabajo estándar de GIT se basa en que los archivos pasan por varios lugares antes de validarse en el repositorio. Estos lugares son:

- El **directorio de trabajo o el área de working**, que es el directorio local de trabajo, donde creamos archivos, y realizamos todas las modificaciones. Cuando se realiza una modificación de un archivo pasa al área de working.
- El **área de staging o preparación**, es un estado intermedio en el que se prepara a los archivos para luego pasar al área de commit. Por tanto, los cambios de los archivos del área de working, antes de validarse y pasar al repositorio local o remoto, pasarán por el área de preparación.
- El **repositorio local**, este repositorio lo creamos con GIT, y es donde se guardan los archivos con las modificaciones y todas las versiones por las que ha pasado. Normalmente pasaremos los archivos del área de preparación, al área de commit.
- El **repositorio remoto**, este repositorio estará fuera del equipo local, que, en el caso de colaborar con más usuarios, podremos sincronizar nuestros ficheros y cambios realizados con el repositorio remoto. Git nos va a permitir acceder a los repositorios remotos desde cualquier sitio con internet.

Archivos rastreados y sin rastrear

Cada archivo de tu repositorio puede tener dos estados: **rastreados y sin rastrear**. Los archivos rastreados (tracked files) son todos los que estaban en la última instantánea proyecto; pueden ser archivos sin modificar, modificados o preparados.

Los archivos sin rastrear son todos los demás – cualquier otro archivo en tu directorio de trabajo que no estaba en tu última instantánea y que no está en el área de preparación (staging area)-. Cuando clonas por primera vez un repositorio, todos tus archivos están rastreados y sin modificar.

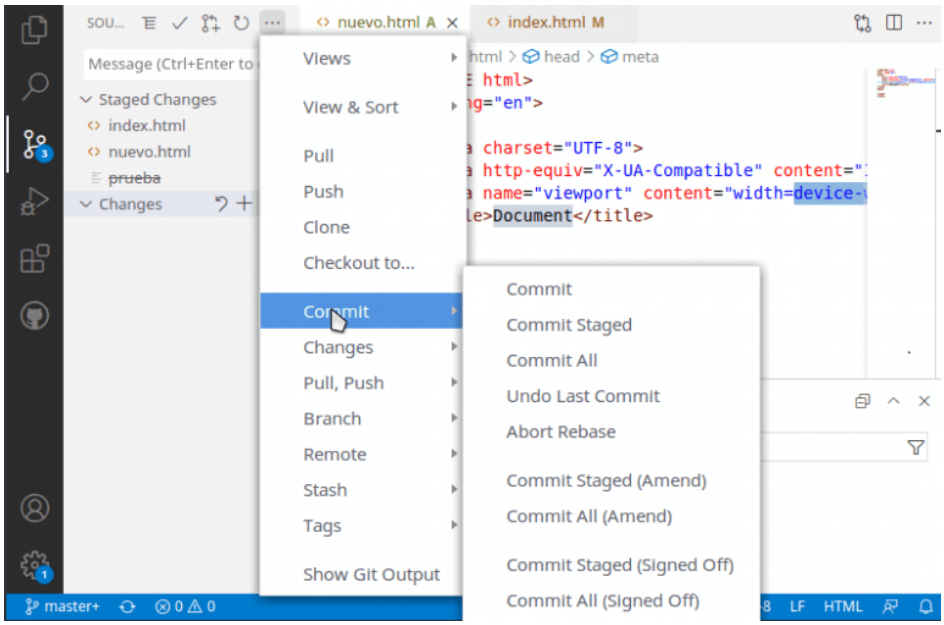
Mientras editas archivos, Git los ve como modificados. Luego preparas estos archivos modificados (staged) y finalmente confirmas todos los cambios preparados (commit).

git-scm.com/book/es/v2/Fundamentos-de-Git-Guardando-cambios-en-el-Repositorio

13.2.5. Stage, commit, push y pull

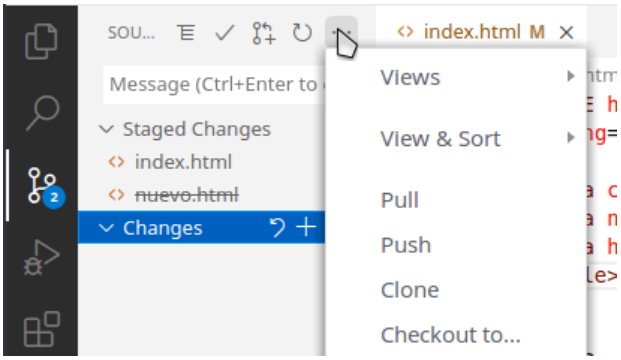
Para hacer "commit" de los ficheros. En primer lugar, incluiremos los ficheros a los que queremos hacer **stage (preparar los ficheros para confirmación y seguimiento)** pulsando en "+" en cada uno de los ficheros o en "todos".

A continuación, si pulsamos en el **icono de tres puntos** nos sale un desplegable con diferentes opciones para hacer **push (subir al servidor)**, **pull (bajar del servidor), commit**... Por supuesto, cuando hagamos «commit» nos pedirá el mensaje identificativo de dicho commit.

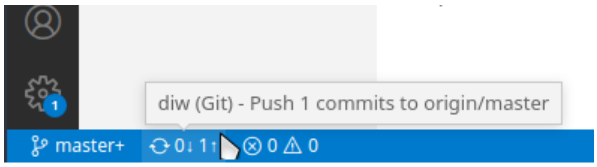


Para hacer **push** de los cambios realizados o **pull** para copiar los datos del repositorio remoto en nuestra computadora podemos hacerlo de dos maneras:

- Podemos pulsar en la parte derecha de la barra de git (icono tres puntos) y elegir la opción *push* o *pull*.



- Podemos pulsar en la parte inferior sobre sincronizar (en este caso haríamos *push* y *pull*).



> El comando `git pull` se emplea para extraer y descargar contenido desde un repositorio remoto y actualizar al instante el repositorio local para reflejar ese contenido. El comando `git push` se utiliza para subir los cambios realizados en local al repositorio remoto.

13.2.6. Realizar los mismos pasos mediante línea de comandos

Clonar el repositorio de GitHub en local

Mediante línea de comandos nos vamos a la carpeta en la que queremos clonar el proyecto.

```
$ cd C:\Users\Andrea\Desktop\Mis proyectos Git\
```

Ahora clonamos el repositorio pegando la URL del repositorio. Por defecto va a clonar la rama "main".

```
$ git clone https://github.com/DIW2122/tenda.git
```

```
PS C:\Users\Andrea\Desktop\Mis proyectos Git> git clone https://github.com/DIW2122/tenda.git
Cloning into 'tenda'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 583 bytes | 53.00 KiB/s, done.
PS C:\Users\Andrea\Desktop\Mis proyectos Git>
```

Si revisamos la carpeta "Mis proyectos Git" veremos que se ha clonado el repositorio.

Guardar cambios en el repositorio local (commit)

Vamos a realizar algún cambio sobre el proyecto. Por ejemplo modificamos el archivo readme.md y guardamos. Ahora avisamos a Git de que queremos preparar los ficheros para

hacer seguimiento -stage- (git add) y un commit (git commit, este commit guarda los cambios en el repositorio local).

```
$ git add --all
$ git commit -am "readme.md actualizado"
```

```
PS C:\Users\Andrea\Desktop\Mis proyectos Git\tenda> git commit -am "readme.md actualizado"
[main a07757d] readme.md actualizado
 1 file changed, 1 insertion(+), 1 deletion(-)
PS C:\Users\Andrea\Desktop\Mis proyectos Git\tenda>
```

Subir cambios locales al repositorio remoto (push)

A continuación vamos a subir los cambios al repositorio remoto de GitHub:

```
$ git push
```

```
PS C:\Users\Andrea\Desktop\Mis proyectos Git\tenda> git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 257 bytes | 257.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/DIW2122/tenda.git
   3052f63..a07757d  main -> main
PS C:\Users\Andrea\Desktop\Mis proyectos Git\tenda>
```

De esta manera hemos subido los cambios locales a GitHub.

Ver resumen de comandos Git

POSIBLE ERROR A SOLUCIONAR: ***fatal: not a git repository (or any of the parent directories): .git»***

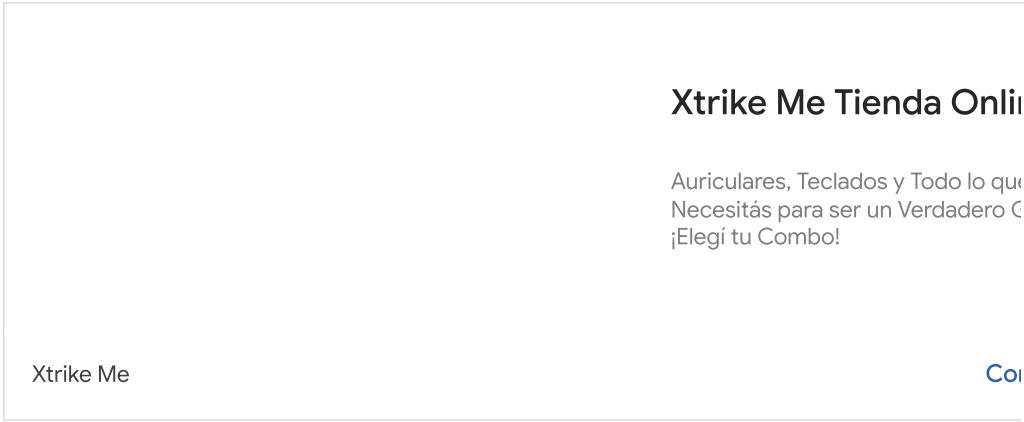
Un posible error que te puede surgir es que la carpeta en la que estás situado no es la raíz con los archivos del proyecto. En ese momento debes hacer un cambio de directorio para situarte en la raíz. Observa el siguiente ejemplo en el que se muestra el error: ***«fatal: not a git repository (or any of the parent directories): .git»*** cuando no estás en la raíz del proyecto «tenda».

```
PS C:\Users\Andrea\Desktop\Mis proyectos Git> git commit -am "readme.md actualizado"
fatal: not a git repository (or any of the parent directories): .git
PS C:\Users\Andrea\Desktop\Mis proyectos Git> cd tenda
PS C:\Users\Andrea\Desktop\Mis proyectos Git\tenda> git commit -am "readme.md actualizado"
[main b0dfa59] readme.md actualizado
 1 file changed, 1 insertion(+), 1 deletion(-)
PS C:\Users\Andrea\Desktop\Mis proyectos Git\tenda> git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 264 bytes | 264.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/DIW2122/tenda.git
   a07757d..b0dfa59  main -> main
PS C:\Users\Andrea\Desktop\Mis proyectos Git\tenda>
```

13.3. Ver los cambios en Github

En GitHub dale a “Settings / Pages y selecciona en el apartado Source “**Main**”. Además, puedes seleccionar un «theme». A continuación, guarda los cambios.
Ahora ya puedes visualizar tu web. La URL de visualización tendrá el siguiente aspecto: **https://nombredetucuenta/github.io/nombredeturepositorio/** (La página mostrada es la página index.html). En mi caso es la siguiente: https://eniun.github.io/diw/

> Si quieres seguir aprendiendo funcionalidades de Github puedes comenzar haciendo el siguiente tutorial: guides.github.com/activities/hello-world/.



13. 4. Otros errores en la configuración de Github con Visual Studio Code

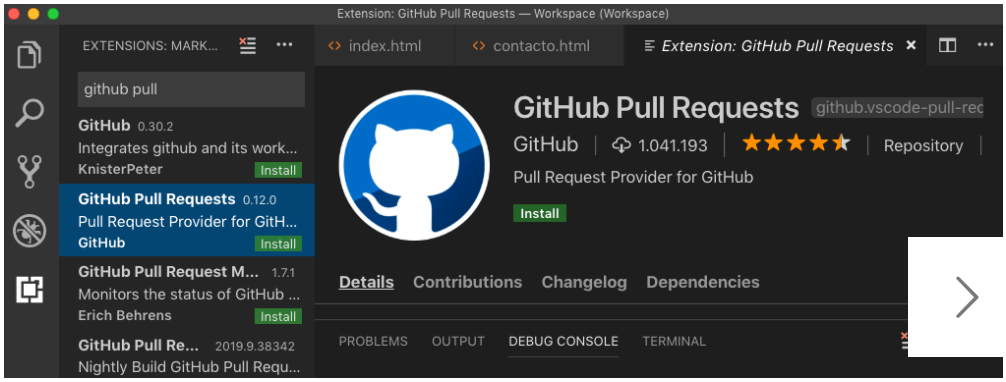
13.4.1. Error: command ‘git.clone’ not found

Para solucionar el error command ‘git.clone’ not found debemos instalar GIT e indicar por línea de comandos nuestro usuario e email:

```
$ git config --global user.name eniun
$ git config --global user.email info@eniun.com
```

13.4.2. Error al hacer pull

Puede resultar necesario instalar la extensión “GitHub pull request” desde el menú *extensions*.



Si los errores persisten puede que te interese utilizar Github con SourceTree para publicar tu código mediante repositorios.

14. Crear organizaciones y grupos

En el siguiente tutorial conocerás qué son las organizaciones y los grupos de GitHub. Además aprenderás a crear organizaciones y a invitar miembros a los grupos creados. Ver el tutorial: eniun.com/crear-organizaciones-equipos-repositorios-github/

Compartir