

# INSTALENNN

GIT

<https://git-scm.com/>



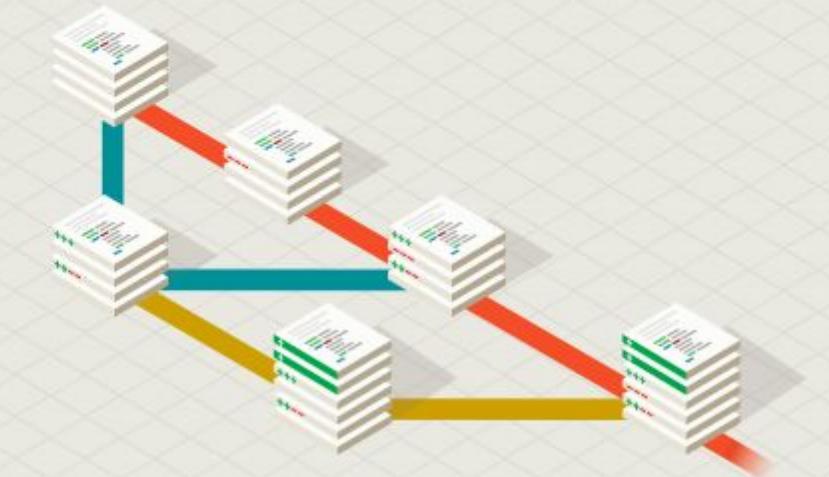
git --local-branching-on-the-cheap

Type / to search entire site...



Git is a [free and open source](#) distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is [easy to learn](#) and has a [tiny footprint with lightning fast performance](#). It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like [cheap local branching](#), [convenient staging areas](#), and [multiple workflows](#).



## About

The advantages of Git compared to other source control systems.



## Documentation

Command reference pages, Pro Git book content, videos and other material.



## Downloads

GUI clients and binary releases for all major platforms.



## Community

Get involved! Bug reporting, mailing list, chat, development and more.



# ¡Javascript!

guayerd beta hub

# Agenda del día



## 01

### Introducción

- Repaso Flexbox.
- Repaso RWD.
- Repaso Bootstrap.

## 02

### GIT

- Definición.
- Versión.
- Configuración.
- Repositorio.
- Estados de repositorio.
- git add
- git commit
- git status
- git log

## 03

### Ejercitación

- Creación del repositorio local de nuestro proyecto.



# daily

¿Cómo venimos?

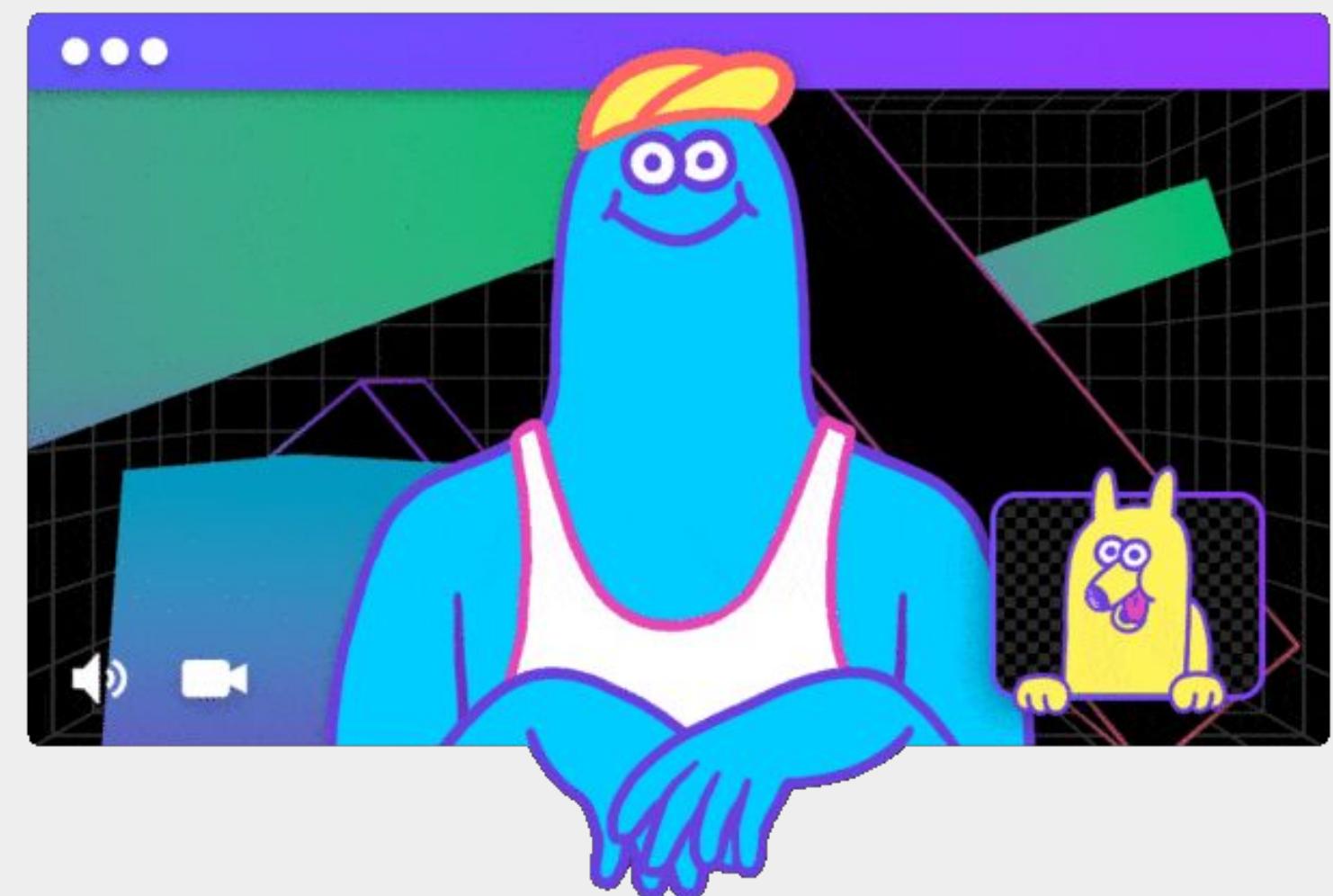
¿Algo nos bloquea?

¿Cómo seguimos?



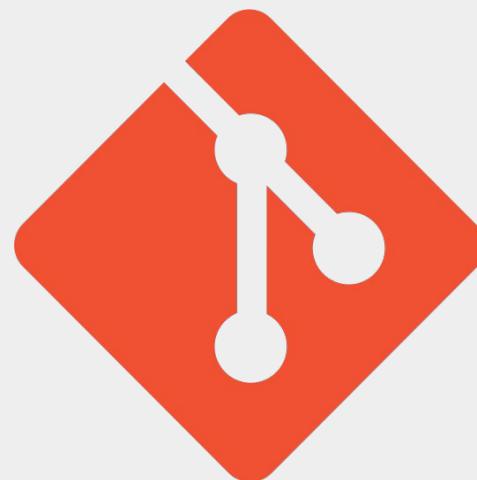
# GIT

¿Qué es?



## SPRINT 1

# Git



# git

**Git es un sistema de control de versiones distribuido, gratuito y de código abierto.**

**Permite administrar y proteger la línea de tiempo de nuestro proyecto y crear múltiples versiones del mismo.**

# Git

Todo cambio en nuestra aplicación web que sea confirmado en Git podrá recuperarse.

## En caso de fuego 🔥

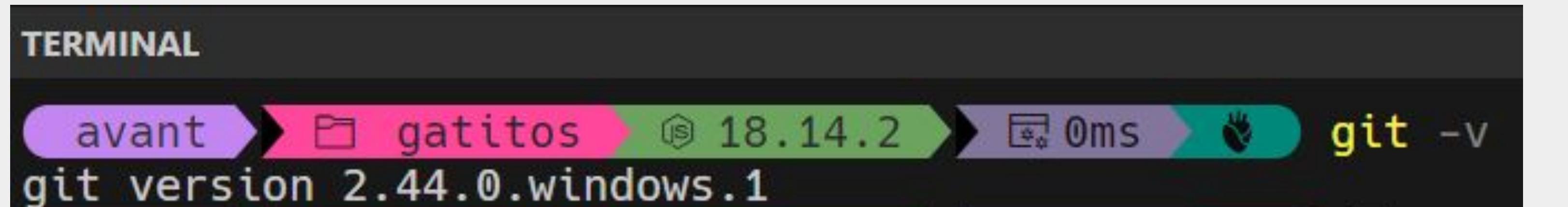
1.  git commit

2.  git push

3.  salga del edificio

# Chequear si Git está instalado

Si el comando `git -v` nos retorna un número de versión tendremos Git instalado.



The screenshot shows a terminal window with a dark background and light-colored text. The title bar says "TERMINAL". The command `git -v` was run, and the output is displayed in green text: `git version 2.44.0.windows.1`. The terminal interface includes colored arrows and icons between the command and its output.

# Configurar usuario global

Es importante que el email que escribamos no sea real, para evitar **SPAM**.



# Ver archivo de configuración global

La configuración global de GIT está en Windows en un archivo ubicado en:

**C:\Users\[username]\.gitconfig**



```
1 [user]
2   email = monkey@donkey.com
3   name = Patos
4

SALIDA PUERTOS PROBLEMAS TERMINAL
TERMINAL
avant ➔ 🗂 ~ ➔ 0ms ➔ 🌐 git config --list
user.email=monkey@donkey.com
user.name=Patos
```

The screenshot shows a terminal window with the following content:

```
1 [user]
2   email = monkey@donkey.com
3   name = Patos
4

SALIDA PUERTOS PROBLEMAS TERMINAL
TERMINAL
avant ➔ 🗂 ~ ➔ 0ms ➔ 🌐 git config --list
user.email=monkey@donkey.com
user.name=Patos
```

The terminal window has tabs for SALIDA, PUERTOS, PROBLEMAS, and TERMINAL. The TERMINAL tab is active. Below the tabs, it says TERMINAL. The command `git config --list` was run, and its output is displayed: `user.email=monkey@donkey.com` and `user.name=Patos`.

# Repositorio

¿Qué es?



SPRINT 1

# Repositorio



**Un repo Git es un almacén donde se guarda todo el código de un proyecto, junto a su historial de cambios.**

**Permite alojar todas las versiones del proyecto para que no se pierda nada.**

# Ventajas de usar repos

- Promueve un código organizado.
- Facilita la colaboración, el trabajo de desarrollo en equipo y remoto.
- Permite ir a cualquier punto del historial del proyecto.
- Ayuda a crear distintas versiones de nuestra aplicación.
- Evita la pérdida de archivos y horas de trabajo.
- Es un manifiesto que nos permite ver el pasado para ir mejorando nuestro código.

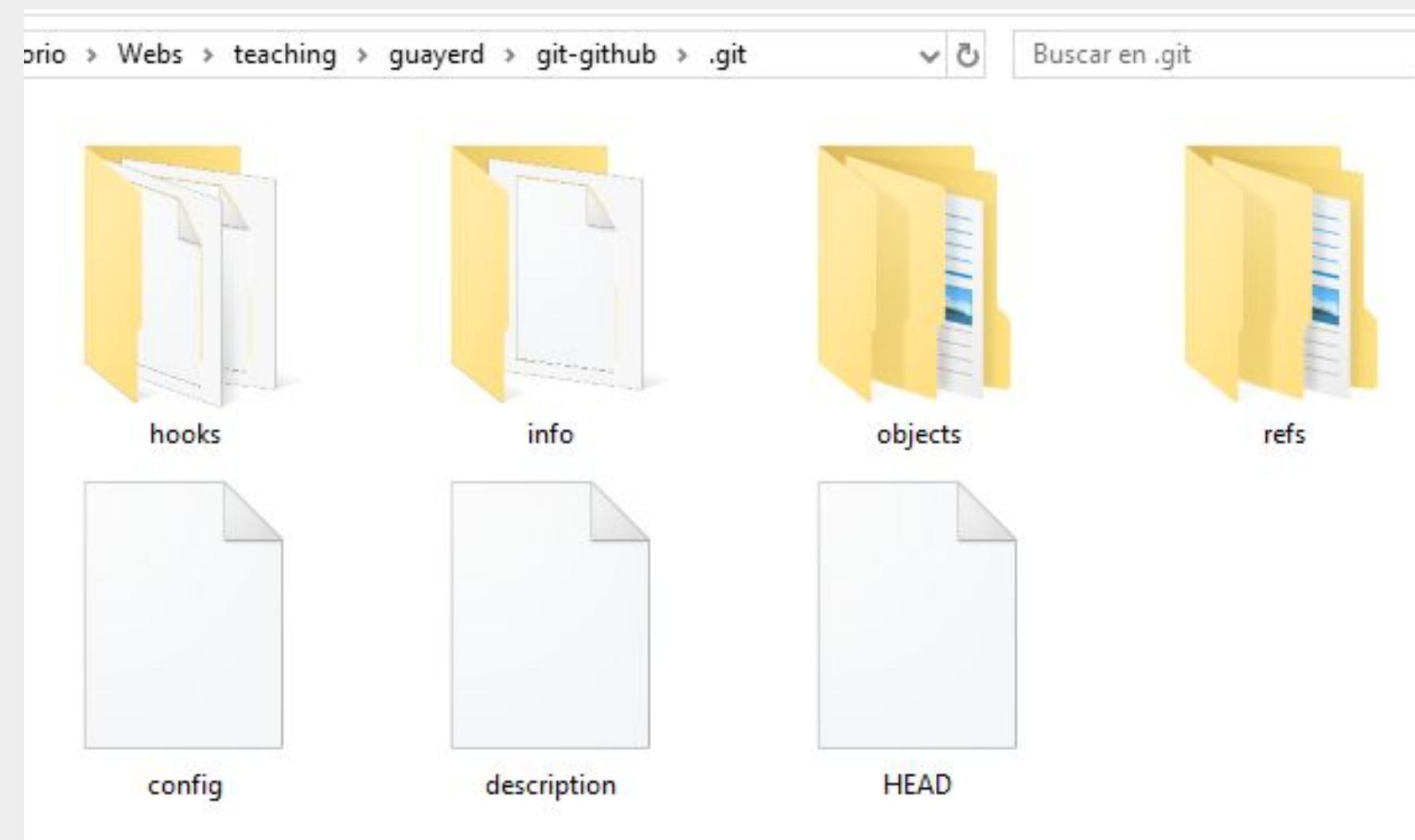
# Crear un repo con git init

Creamos una carpeta .git pero no la vemos en la columna izquierda de Visual Code.

The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar displays a folder named "GIT-GITHUB" containing an "index.html" file. The main editor area shows the content of "index.html". Below the editor, tabs for "SALIDA", "PUERTOS", "PROBLEMAS", and "TERMINAL" are present, with "TERMINAL" being the active tab. The terminal window at the bottom shows the command "git init" being run in a directory named "git-github", which is highlighted in pink. The output of the command, "Initialized empty Git repository in C:/Users/avant/OneDrive/Escritorio/Webs/teaching/guayerd/git-github/.git/", is also visible in the terminal window.

# Ver el repo creado

En el explorador de Windows.



# Ver estado del repo

Con git status podemos ver el estado actual del repositorio.

The screenshot shows the VS Code interface. In the Explorer sidebar, there is a folder named 'GIT-GITHUB' containing a file 'index.html'. The file content is:

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8" />
5     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6     <title>Javascript</title>
7   </head>
8   <body></body>
9 </html>
```

In the Terminal tab, the command 'git status' is run, showing:

```
avant ➜ git-github ➜ master ✘?1 ➜ 0ms ➜ git status
On branch master

No commits yet

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

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

- **No commits yet** indica que no tenemos commits.
- **Untracked files** muestra archivos que Git no detecta (se ven en rojo).

# 3 estados de un repo

## Working Directory

Son archivos, los cuáles no están siendo seguidos por Git o que son seguidos pero que no han sido guardados sus últimos cambios.

**Se ven en rojo.**

## Staging Area

Archivos que están listos para ser guardados en su versión actual.

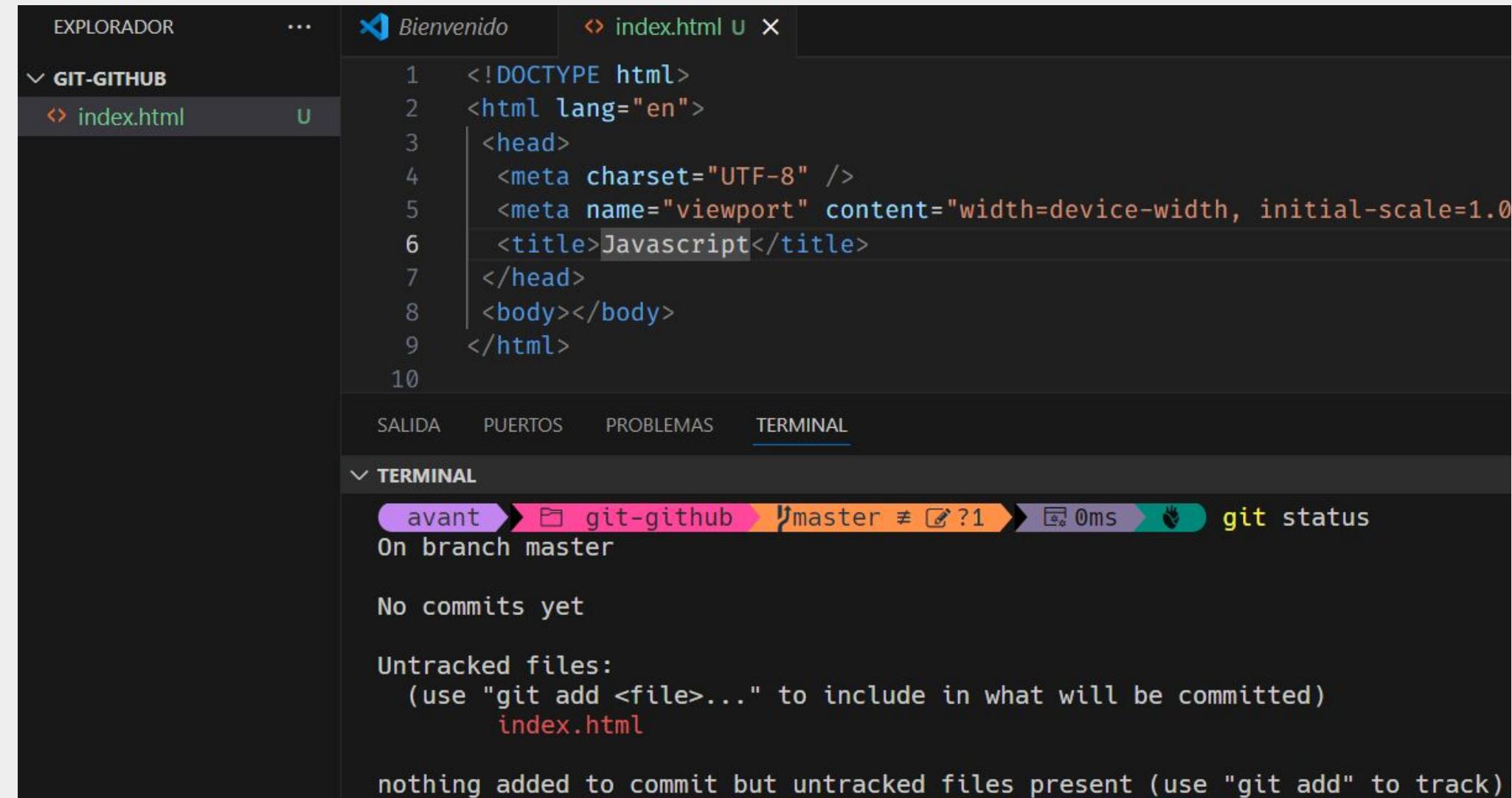
**Se ven en verde.**

## Repository

Archivos confirmados en el repo que quedan registrados en la línea de tiempo de Git.

# Working directory

Los archivos en working directory se ven en color **rojo**. Para seguir sus últimos cambios o agregarlo si es un nuevo archivo al staging area debemos ejecutar **git add [file]**



The screenshot shows the VS Code interface with the following details:

- EXPLORADOR:** Shows a folder named "GIT-GITHUB" containing "index.html". The "index.html" file is highlighted in red, indicating it is untracked.
- Bienvenido:** Shows the content of "index.html":

```
1  <!DOCTYPE html>
2  <html lang="en">
3  | <head>
4  | <meta charset="UTF-8" />
5  | <meta name="viewport" content="width=device-width, initial-scale=1.0
6  | <title>Javascript</title>
7  | </head>
8  | <body></body>
9  </html>
```
- TERMINAL:** Shows the command "git status" output:

```
avant ➜ git-github ➜ master ✘ ?1 ➜ 0ms ➜ git status
On branch master

No commits yet

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

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

# git add

```
avant ➜ git-github master ✘ ?1 ~2 | ✓ +2 ⟲ 18.14.2 ➤ 0ms ➔ 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:  styles.css

Changes not staged for commit:
(use "git add <file>..." to update what will be committed)
(use "git restore <file>..." to discard changes in working directory)
  modified:  index.html
  modified:  styles.css

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

avant ➜ git-github master ✘ ?1 ~2 | ✓ +2 ⟲ 18.14.2 ➤ 50ms ➔ git add .
avant ➜ git-github master ✘ ?1 ~2 | ✓ +3 ⟲ 18.14.2 ➤ 51ms ➔ git status
avant ➜ git-github master ✘ ?1 ~2 | ✓ +3 ⟲ 18.14.2 ➤ 0ms ➔ 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:  index.js
  new file:  styles.css
```

Podemos agregar, por ejemplo, el archivo `index.html` al staging area con el comando:

**git add index.html**

También podemos agregar todos los archivos en rojo con **git add .**

# git commit

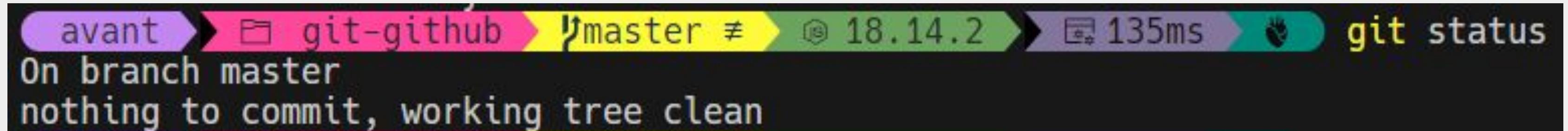
Para pasar archivos en staging area (**verdes**) al estado final repository debemos ejecutar el comando **git commit** incluyendo un mensaje con el flag **-m**.

Hemos creado nuestra primera confirmación (**commit**).

```
avant ➔ git-github ➔ master ✘ +3 ➔ 18.14.2 ➔ 0ms ➔ git commit -m "primer commit"
[master (root-commit) 4a1487f] primer commit
 3 files changed, 11 insertions(+)
 create mode 100644 index.html
 create mode 100644 index.js
 create mode 100644 styles.css
```

# Ver estado del repo

Con git status podemos ver el estado actual del repositorio. En este caso indica que el árbol está limpio.



```
avant ➔ git-github ➔ master ✘ 18.14.2 ➔ 135ms ➔ git status
On branch master
nothing to commit, working tree clean
```

A screenshot of a terminal window with a dark background and colored tabs. The tabs are purple, pink, yellow, and green. The text in the terminal shows the user is on the 'master' branch and there is nothing to commit, indicating a clean working tree.

# Ver historial del repo

¿Para qué hemos realizado todo esto? Hemos creado un nuevo punto en el historial.

Usemos **git log** para revisar.

Observamos que nuestro usuario con nuestro mail creó un commit identificado con el número 4a1487f7ffd0603838caf22ea980978d93d9e058 en una fecha con el mensaje “primer commit”.

```
avant ➔ git-github ➔ master ≠ ➔ 18.14.2 ➔ 48ms ➔ git log  
commit 4a1487f7ffd0603838caf22ea980978d93d9e058 (HEAD -> master)  
Author: Pato <monkey@donkey.com>  
Date:   Sun Jul 21 17:40:16 2024 -0300  
  
        primer commit
```

# Probemos cosas

Arranquemos acá:

```
└─ GIT-GITHUB
    └─ index.html
    └─ index.js
    └─ # styles.css
```

Llegamos a esto:

- 1) Borremos el archivo styles.css.
- 2) git add .
- 3) git commit -m "Borré styles.css"
- 4) git log

```
└─ GIT-GITHUB
    └─ index.html
    └─ index.js
```

# Probemos cosas

Ahora tenemos 2 commits.

```
avant ➔ git-github ✘ master ≠ ↗ 18.14.2 ➤ 0ms ➔ git add .
avant ➔ git-github ✘ master ≠ ✓ -1 ↗ 18.14.2 ➤ 44ms ➔ git commit -m "Borré styles.css"
[master 02540d7] Borré styles.css
  1 file changed, 2 deletions(-)
  delete mode 100644 styles.css
avant ➔ git-github ✘ master ≠ ↗ 18.14.2 ➤ 104ms ➔ git log
commit 02540d7b92dca5869b5177897b75cfdb9ea32646 (HEAD -> master)
Author: Pato <monkey@donkey.com>
Date:  Sun Jul 21 17:55:18 2024 -0300

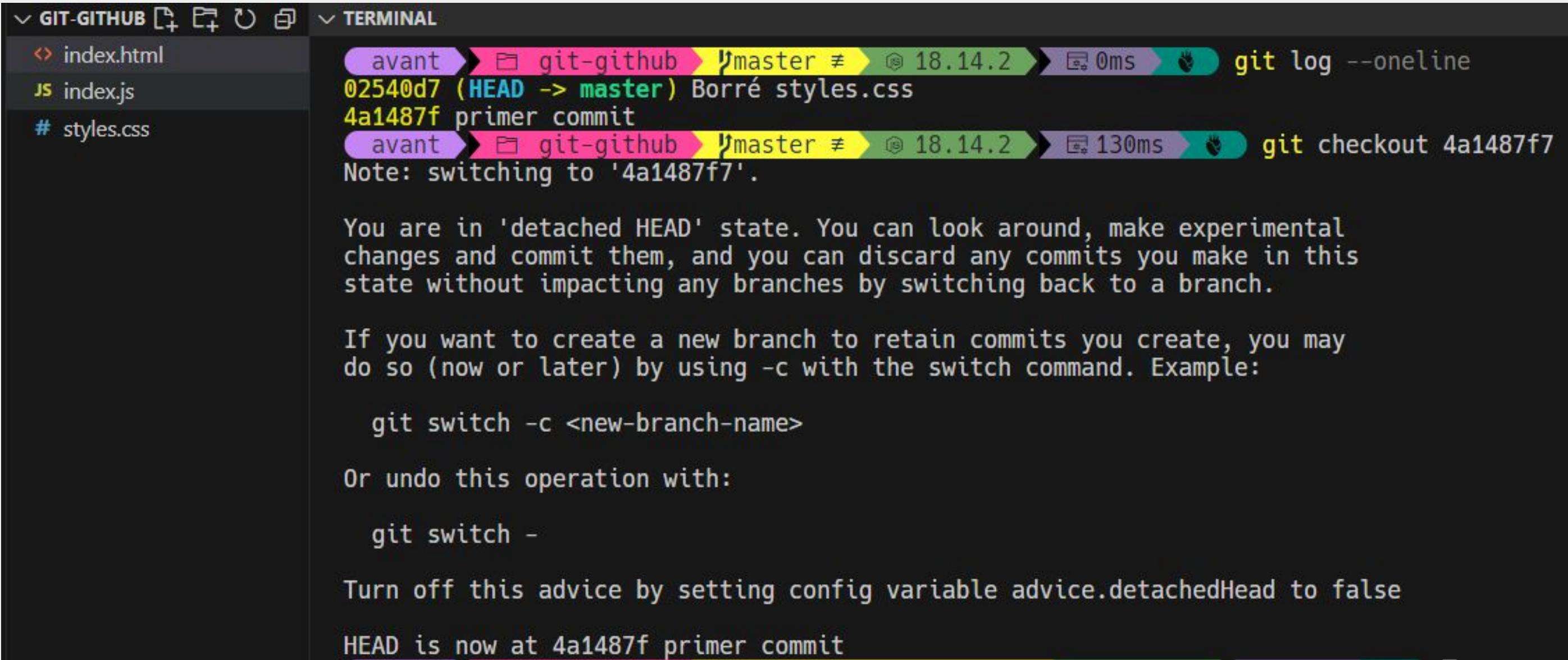
  Borré styles.css

commit 4a1487f7ffd0603838caf22ea980978d93d9e058
Author: Pato <monkey@donkey.com>
Date:  Sun Jul 21 17:40:16 2024 -0300

  primer commit
```

# Volvamos al commit anterior

Debemos usar el comando **git checkout idcommit**



```
git log --oneline
02540d7 (HEAD -> master) Borré styles.css
4a1487f primer commit
git checkout 4a1487f7
Note: switching to '4a1487f7'.
```

You are in 'detached HEAD' state. You can look around, make experimental changes and commit them, and you can discard any commits you make in this state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may do so (now or later) by using -c with the switch command. Example:

```
git switch -c <new-branch-name>
```

Or undo this operation with:

```
git switch -
```

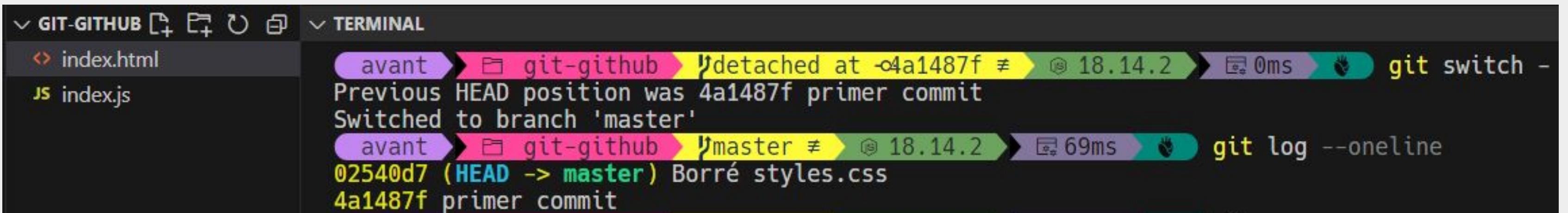
Turn off this advice by setting config variable advice.detachedHead to false

```
HEAD is now at 4a1487f primer commit
```

# Volvamos al último commit

Debemos usar el comando **git switch -**

Se volvió a ir styles.css.

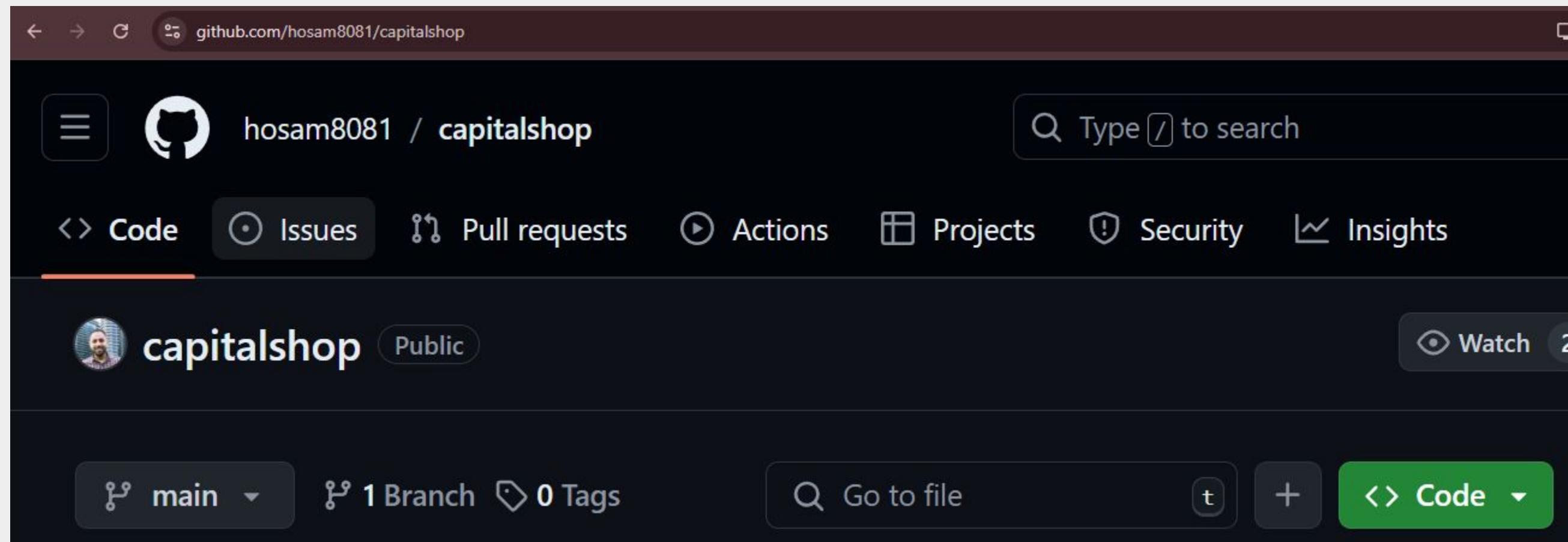


The screenshot shows a terminal window with a dark theme. On the left, there's a sidebar labeled 'GIT-GITHUB' with files 'index.html' and 'index.js'. The main area is labeled 'TERMINAL' and contains the following command history:

```
avant ➔ git-github ➔ ↵detached at 4a1487f ≠ ➔ 18.14.2 ➔ 0ms ➔ git switch -
Previous HEAD position was 4a1487f primer commit
Switched to branch 'master'
avant ➔ git-github ➔ ↵master ≠ ➔ 18.14.2 ➔ 69ms ➔ git log --oneline
02540d7 (HEAD -> master) Borré styles.css
4a1487f primer commit
```

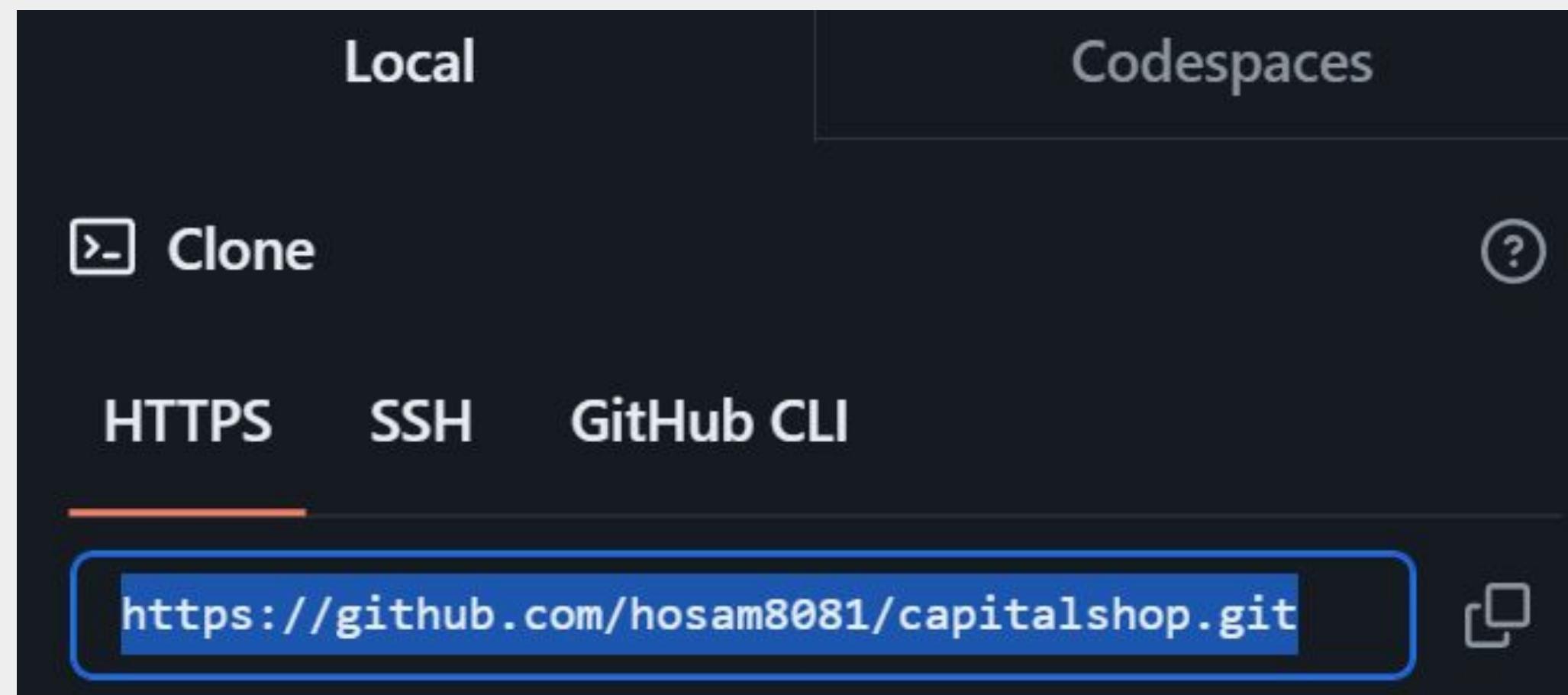
# Clonar un repositorio de GitHub

Vamos al repo y damos clic al botón **verde**.



# Clonar un repositorio de GitHub

Copiamos la dirección del repo (**texto azul**).



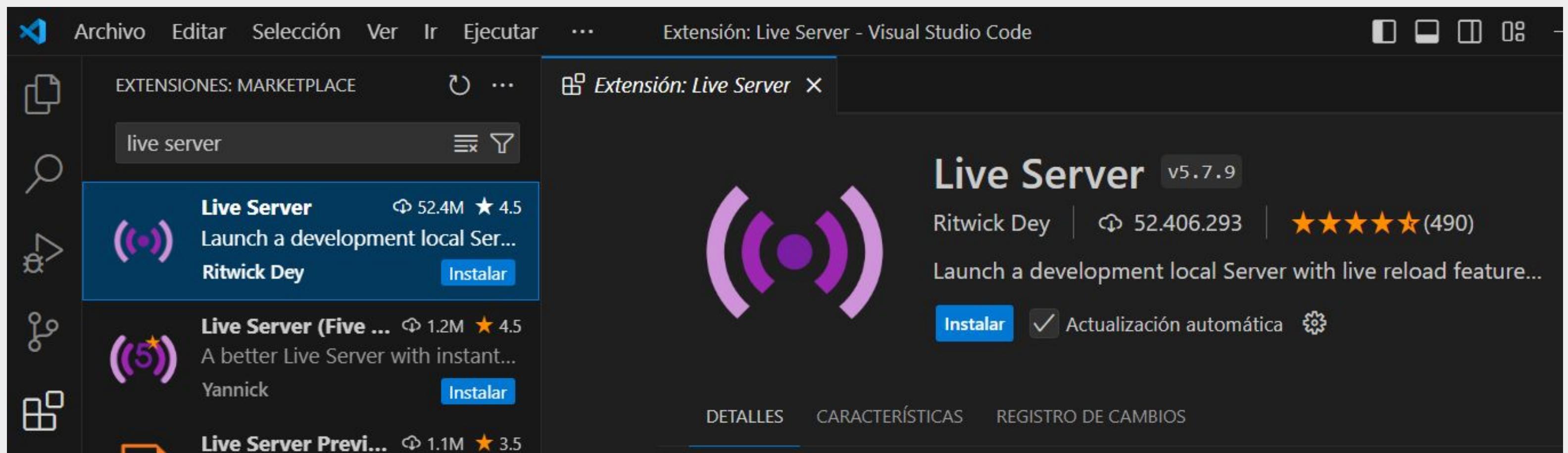
# Clonar un repositorio de GitHub

Debemos usar el comando **git clone [reponame]** en la terminal.

```
✓ TERMINAL
  avant ➔ 📁 guayerd ➔ 🏃 0ms ➔ ⌐ git clone https://github.com/hosam8081/capitalshop.git
  Cloning into 'capitalshop'...
  remote: Enumerating objects: 130, done.
  remote: Counting objects: 100% (130/130), done.
  remote: Compressing objects: 100% (110/110), done.
  remote: Total 130 (delta 54), reused 81 (delta 19), pack-reused 0 (from 0)
  Receiving objects: 100% (130/130), 1.30 MiB | 3.24 MiB/s, done.
  Resolving deltas: 100% (54/54), done.
```

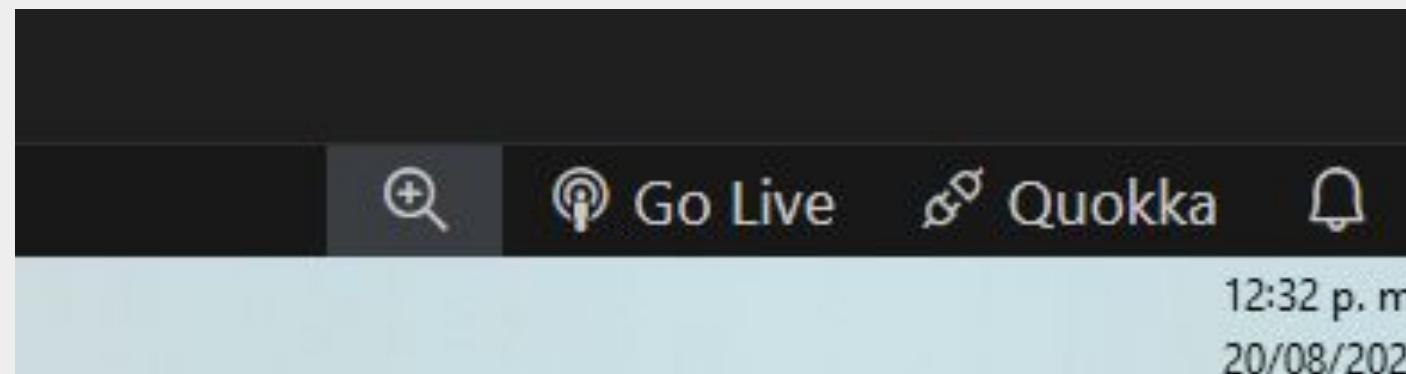
# Instalar Live Server

Vamos a extensiones en la barra lateral izquierda y en el buscador escribimos **Live Server**. Lo seleccionamos y lo instalamos. Live Server funciona con Hot Reloading.



# Live Server

Aparecerá ahora en la parte inferior derecha. Damos clic en **Go Live**.



 Capital Shop

Home shop blog contact 

Login Sign Up

Sale Up To 50% Biggest Discounts. Hurry! Limited Perriod Offer ShoP Now



*fashoin sale*

## minmal menz style

Consectetur adipisicing elit. Laborum fuga  
incident laboriosam voluptas iure, delectus  
dignissimos facilis neque nulla earum.

Buy Now

## En caso de fuego 🔥



1.git commit



2.git push



3.salga del edificio



¡Manos a la obra!

SPRINT 1

# Ejercicio

Clonar un repositorio.

1.

Clonar este repositorio:

<https://github.com/hosam8081/capitalshop>

2.

Hacerlo andar en el servidor de desarrollo con Go Live.

SPRINT 2

# Ejercicio

Crear repositorio local.

1.

Crear un repositorio en la carpeta de nuestro:

`e-commerce_apellido_nombre`

2.

Confirmar (commit) todos los cambios.

# 🔍 retro

¿Cómo nos fué?

¿Qué cosas no quedaron claras y  
necesitamos repasar la próxima?

