



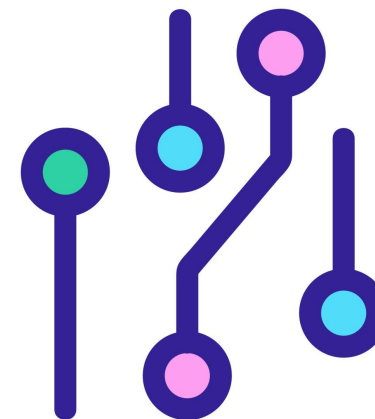
# Introducción a Git & Github

Control de versiones y colaboración en desarrollo Frontend

**ABRIMOS RUTAS INCLUSIVAS  
AL TALENTO DIGITAL**

# Objetivos generales y contenidos del taller

- Descubrir que es git y para qué sirve
- Poner en práctica los comandos git
- Trabajar en equipo en Github



# ¿Qué es Git?


## ¿Qué es Git?

- **Definición:** Git es un sistema de control de versiones que permite rastrear los cambios en el código, trabajar en equipo y revertir errores.
- **¿Por qué usar Git?**
  - Mantener un historial de versiones del proyecto.
  - Trabajar en equipo sin conflictos.
  - Probar nuevas ideas sin afectar la versión estable.
- **Ejemplo:**
  - Una diseñadora Frontend desarrolla un botón. Con Git, puede guardar (commit) el estado del código antes de probar un cambio, y volver atrás si algo sale mal.



# Palabras clave de Git

---



**Git bash** : Es un emulador de terminal que te permite ejecutar comandos de Git desde una interfaz de línea de comandos

**Git GUI** : Es una interfaz gráfica de usuario para Git que te permite realizar operaciones de Git con el mouse

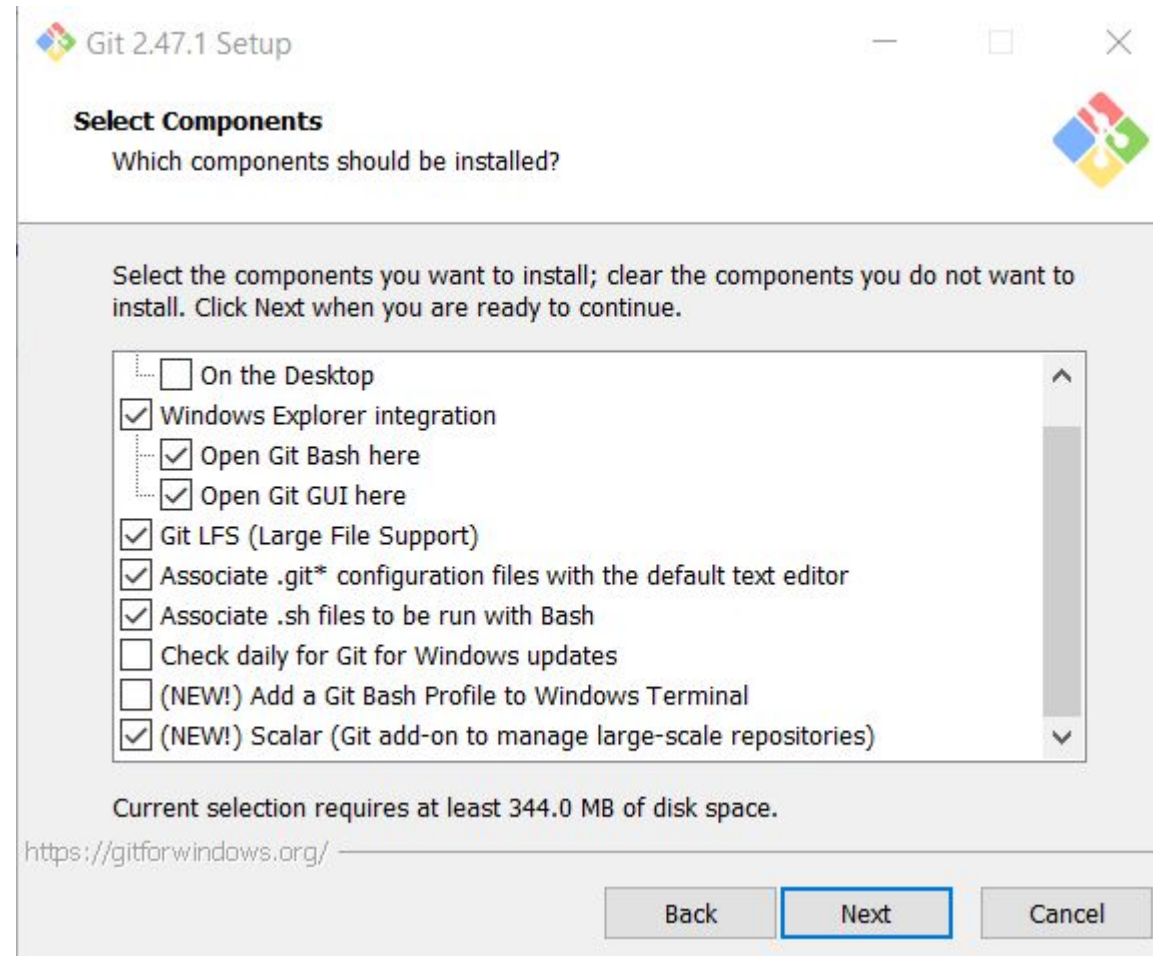
**Git LFS** : es una extensión de Git que te permite almacenar archivos grandes en un servidor remoto

**Terminal de Windows**: Es una aplicación de consola que te permite ejecutar comandos de Windows



# Instalación de Git en Windows

Importante tener  
seleccionadas estas  
opciones



# Instalación de Git en Mac

---



[https://www.youtube.com/watch?v=2CT\\_OZSOTpQ](https://www.youtube.com/watch?v=2CT_OZSOTpQ)

<https://www.youtube.com/watch?v=5EXAugHfhuk>



# Conceptos básicos de Git

---

- Repositorio: Es un "contenedor" de tu proyecto, que incluye todos los archivos y el historial de cambios.
- Commit: Un "punto de control" en el historial del proyecto.
- Branch (rama): Una línea de desarrollo paralela.
- Merge: Combinar ramas.
- Staging area: Espacio intermedio antes de confirmar los cambios.
- Ejercicio práctico:
  - Crear un repositorio y guardar tu primer commit (más adelante se detalla).



# ¿Qué es GitHub?

Definición: Una plataforma basada en la nube para almacenar, compartir y colaborar en proyectos Git.

¿Por qué usar GitHub?

- Backup en la nube.
- Colaboración en tiempo real.
- Publicar proyectos en portafolios.

Ejemplo: Un equipo desarrolla una aplicación web. Cada miembro tiene acceso al mismo repositorio en GitHub, lo que permite una integración fluida de los cambios.





# Instalación y configuración de Git

## Descargar e instalar Git:

- Ir a [git-scm.co](https://git-scm.co) y descargar la versión adecuada para tu sistema operativo.

## Configurar Git:

bash

Copiar código

```
git config --global user.name "Tu Nombre"
```

```
git config --global user.email "tuemail@example.com"
```

- Esto asocia tu identidad a los commits.

<https://www.youtube.com/watch?v=8g-ElbPJmGA>



# Flujo básico de Git

## Clonar o inicializar un repositorio:

- Clonar: `git clone <URL-del-repositorio>`
- Inicializar: `git init`

## Agregar cambios:

bash

Copiar código

`git add <archivo> # Añade un archivo específico`

`git add . # Añade todos los cambios`

## Confirmar los cambios:

bash

Copiar código

`git commit -m "Mensaje descriptivo"`

## Subir al repositorio remoto (GitHub):

bash

Copiar código

`git push origin main`



# Flujo de trabajo colaborativo

Crear una rama:

bash

Copiar código

`git checkout -b nombre-de-la-rama`

- Ejemplo: `git checkout -b feature-navbar`.

2. Trabajar y confirmar los cambios.

3. Hacer un merge:

- Cambiar a la rama principal: `git checkout main`.
- Combinar: `git merge nombre-de-la-rama`.

4. Resolver conflictos si los hay.



# Ejercicio práctico

## 1. Crear un repositorio en GitHub:

- Ve a [GitHub](#) y crea un repositorio nuevo.

## Inicializar Git localmente:

bash

Copiar código

`git init`

## 2. Vincular el repositorio remoto:

bash

Copiar código

`git remote add origin <URL-del-repositorio>`

## 3. Realizar cambios y subirlos:

bash

Copiar código

`echo "Hola Mundo" > index.html`

`git add .`

`git commit -m "Primer commit"`

`git push -u origin main`



# Buenas prácticas

---



## Usar mensajes claros en los commits:

- Evitar: "Cambio"
- Usar: "Corrige el diseño del botón en la navbar"

**Trabajar en ramas para cada funcionalidad.  
Realizar revisiones de código antes de fusionar.**



Recursos:

Documentación oficial de Git

Guía rápida de GitHub

Tutoriales interactivos:

<https://learngitbranching.js.org/>





**Somos  
F5**

[www.somosf5.org](http://www.somosf5.org)

Síguenos en

