



# Terminal, Git, GitHub y GitHub Pages

Trabajando con Git y GitHub

***Crear un repositorio remoto en Github para controlar las versiones de un proyecto y publicar la página web utilizando Github Pages.***

- Unidad 1: HTML y CSS
- Unidad 2: Bootstrap
- Unidad 3: JavaScript
- Unidad 4: Terminal, Git, GitHub y GitHub Pages



Te encuentras aquí



## ¿Qué aprenderás en esta sesión?

- *Aplicar el procedimiento de habilitación de una página web con Github Pages, para que el contenido esté disponible públicamente.*

**`/* Git branch */`**

# Git branch

## *Conociendo las ramas del proyecto*

- Una de las principales ventajas de git es la utilización de branch (ramas). Primero definiremos una rama simplemente como un camino donde está nuestro código, mientras que la definición técnica se refiere a un branch como un apuntador a donde van los commits que realizamos.
- Cada vez que inicializamos git, de forma automática se genera un branch main, que es donde se guardarán los nuevos commits.

# Git branch

## *Conociendo las ramas del proyecto*

- Al iniciar git en una carpeta, de forma automática se genera la rama main. Podremos conocer las ramas que tiene nuestro proyecto con el comando:

```
git branch
```

- Nos mostrará en consola todos los branch del proyecto. Como solo tenemos una rama, nos aparecerá **main**.

# Git branch

## *¿Cuándo generar una nueva rama?*

- Para crear una nueva branch utilizaremos el siguiente comando:

```
git branch nueva_branch
```

- Una vez creado, no cambiará de forma automática a la nueva rama, sino que nos quedaremos en la rama original. Para cambiarnos debemos hacerlo con el comando:

```
git checkout nueva_branch
```

# Git branch

## *¿Cuándo generar una nueva rama?*

- Si queremos crear una branch y situarnos enseguida en ella debemos ejecutar el siguiente comando:

```
git checkout -b otra_branch
```

- Veamos con el siguiente comando las ramas que hemos creado:

```
git branch
```



**`/* Git stash */`**

# Git stash

- Existe una manera de reservar o “apartar” las modificaciones que estamos haciendo en nuestra rama actual para realizar una tarea emergente y asegurarnos que no perderemos los cambios que hemos realizado, para esto podemos optar por ocupar el comando:

```
git stash
```

# Git stash

- Todos nuestros cambios han desaparecido, sin embargo, lo que sucedió es que fueron diferidos a un área en paralelo de nuestro historial de commits al cual podemos acceder con el siguiente comando:

```
git stash list
```

# Git stash

- Para volver a unir el directorio actual con los cambios que estábamos trabajando y que fueron apartados del commit en el que nos encontrábamos.

```
git stash apply
```

- Este comando es muy útil para desarrollar sin temor a perder nuestras modificaciones, no obstante se recomienda usarlo en casos puntuales donde necesitemos de forma urgente solucionar problemas que fueron detectados mientras estamos trabajando en una nueva versión de nuestro proyecto.

***/\* Git Rebase \*/***

# Git rebase

- Cuando estamos trabajando en un equipo de desarrollo y varios usuarios manipulan el mismo repositorio en paralelo, es normal que se generen muchas ramas y muchos commits que dificultan la lectura del historial general, puesto que habrán modificaciones agrupadas en una rama A que serán unidas a una rama B a partir del último cambio generado.
- Para lograr un historial más limpio con menos ramas y menos commits distribuidos, git nos ofrece el siguiente comando:

```
git rebase otra_branch
```

**`/* GitHub Pages */`**

# GitHub Pages

## *¿Qué es GitHub pages?*

- GitHub pages, renderiza nuestro código de una rama específica en un dominio y espacio dentro de GitHub.
- Esto será de utilidad para poder mostrar nuestro trabajo de forma gratis, sin tener que recurrir a dominios o servidores externos.



# GitHub Pages

## *¿Qué es GitHub pages?*

“GitHub Pages es un servicio de alojamiento de sitio estático que toma archivos HTML, CSS y JavaScript directamente desde un repositorio en GitHub, opcionalmente ejecuta los archivos a través de un proceso de compilación y publica un sitio web.”

Fuente: [Documentación de Github](#).

# Demostración: “Implementando Github pages”



# Sigue los pasos...

- **Paso 1:** Crear un repositorio, para ello le asignaremos el nombre prueba-ghpages

## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?

[Import a repository.](#)

*Required fields are marked with an asterisk (\*).*

Owner \*

Repository name \*

✔ prueba-ghpages is available.

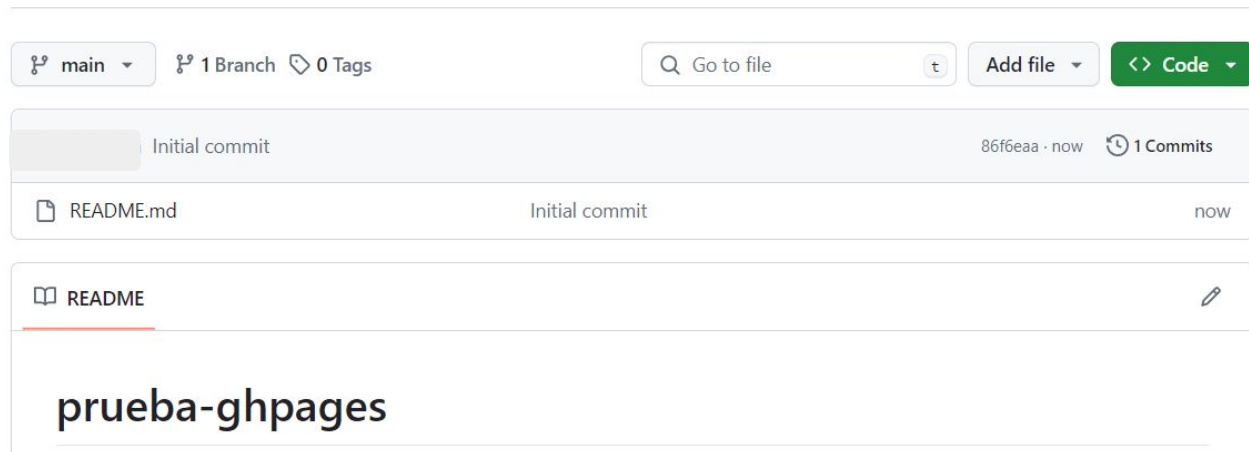
Great repository names are short and memorable. Need inspiration? How about [psychic-pancake](#) ?

Description (optional)



# Sigue los pasos...

## Resultado de creación



The screenshot displays a GitHub repository interface. At the top, the 'main' branch is selected, showing '1 Branch' and '0 Tags'. A search bar labeled 'Go to file' is present, along with 'Add file' and 'Code' buttons. Below this, a commit history section shows an 'Initial commit' with hash '86f6eaa' and timestamp 'now', containing '1 Commit'. A file named 'README.md' is listed with the same commit information. The 'README' tab is active, showing the content 'prueba-ghpages'.

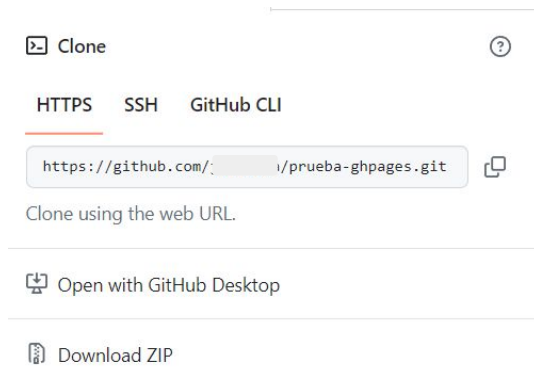


# Sigue los pasos...

- **Paso 2:** Clonamos el repositorio creado de manera local

```
git clone https://github.com/username/username.github.io
```

Recuerda ingresar tu username y el enlace mediante clonación de repositorio.



## Sigue los pasos...

- **Paso 3:** Clonamos el repositorio creado de manera local-
- **Paso 4:** Accedemos al repositorio clonado y lo abrimos en VS Code.
- **Paso 5:** Creamos un archivo index.html y definimos un h1 con el texto "Pruebas de GithubPages"



# Sigue los pasos...

- **Paso 6:** Subimos los cambios a github dado que creamos un documento HTML.

```
git add --all
```

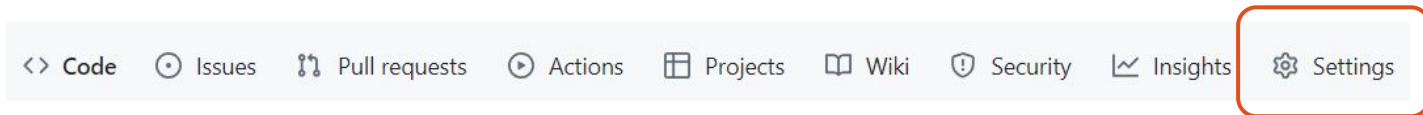
```
git commit -m "Initial commit"
```

```
git push -u origin main
```



## Sigue los pasos...

- **Paso 7:** En el repositorio después de subir el archivo HTML nos dirigimos a Settings o Configuraciones.



Dentro de Settings vamos a buscar en el panel izquierdo la opción Pages





# Visualización de Pages

## GitHub Pages

[GitHub Pages](#) is designed to host your personal, organization, or project pages from a GitHub repository.

## Build and deployment

### Source

Deploy from a branch ▾

### Branch

GitHub Pages is currently disabled. Select a source below to enable GitHub Pages for this repository. [Learn more about configuring the publishing source for your site.](#)

None ▾

Save

## Visibility

GITHUB ENTERPRISE

With a GitHub Enterprise account, you can restrict access to your GitHub Pages site by publishing it privately. A privately published site can only be accessed by people with read access to the repository the site is published from. You can use privately published sites to share your internal documentation or knowledge base with members of your enterprise.

Try GitHub Enterprise risk-free for 30 days

[Learn more about the visibility of your GitHub Pages site](#)



# Sigue los pasos...

## Configuración de Branch

- **Paso 8:** Configuramos la rama, en este caso tenemos una sola por defecto en main. Luego, damos clic en Save.

### Branch

GitHub Pages is currently disabled. Select a source below to enable GitHub Pages for this repository. [Learn more about configuring the publishing source for your site.](#)

 main ▼

 / (root) ▼

Save

# Sigue los pasos...

## Configuración de Branch

- **Paso 9:** Debemos esperar unos segundos o minutos para que se ejecuten los cambios. Una vez realizados, veremos la siguiente información.

### GitHub Pages

[GitHub Pages](#) is designed to host your personal, organization, or project pages from a GitHub repository.

Your site is live at [https://\[username\].github.io/prueba-ghpages/](https://[username].github.io/prueba-ghpages/)

Last [deployed](#) by [avatar] a now

[Visit site](#)

...

Github nos entrega un enlace de acceso.

# Demostración: “Práctica de pull request”



# Contexto

- Pull Request es una acción disponible en Github que permite a un desarrollador solicitar la revisión y aprobación de sus cambios.
- Recordemos en GitHub podemos tener diversas ramas, la principal (main) y alternativas.
- Validar los cambios mediante pull request permitirá ejecutar cambios en una rama distinta a la main y si estos son aprobados, entonces se fusionan a la rama principal.



# Instrucciones

- El/la docente creará un repositorio e invitará a los estudiantes como colaboradores.
- Seleccionar un estudiante que pueda clonar el repositorio y crear una rama distinta a la main.
- Una vez creada la rama por parte del estudiante, se deberá añadir una nueva funcionalidad, esta puede ser un nuevo archivo HTML o alguna hoja de estilos.
- El estudiante deberá subir los cambios a la rama creada para que el/la docente puede revisarlos y entregar feedback.
- Si los cambios son aprobados, el/la docente realiza el merge de la nueva rama.



# Resumen

- En esta unidad hemos aprendido a trabajar con la terminal y sus comandos principales. Además, analizamos la importancia de versionar nuestro código y utilizar recursos tanto locales como remotos para gestionar proyectos.
- Estas herramientas son clave para trabajar desarrollando aplicaciones, pues nos permiten resguardar nuestro progreso, hacer pruebas de concepto sin impactar el código principal y lo que es más relevante aún, trabajar en equipos con responsabilidades diversas.



## Próxima sesión...

- *Revisar material de estudio asincrónico que consiste en un **desafío** para practicar los conceptos aprendidos en esta sesión.*



**{desafío}**  
**latam\_**

*Academia de  
talentos digitales*

