

«Talento Tech»

Git y GitHub

Introducción a GitHub

2



Sesión 2: Introducción a GitHub

Índice

Sesión 2: Introducción a GitHub

Repasso breve de Git

Recordamos el flujo local de Git:

Rol del repositorio remoto

¿Qué es GitHub?

Creación de usuario en GitHub

Verificar instalación

Introducción práctica a GitHub

Conectar el repositorio local con GitHub

Comandos fundamentales:

⚙️ Mini-práctica:

Ramas (branches) y trabajo paralelo

Creando una rama:

Paso a paso para fusionar:

Conflictos

Pull requests y revisión de cambios

Diferencia clave:

Uso inicial de GitHub Desktop (opcional)

Cierre conceptual del encuentro

⚙️ Mini-práctica:

⚙️ Desafío

Simulamos un flujo colaborativo con ramas y pull requests

Objetivos de la clase

En esta sesión continuaremos abordando el conocimiento de Git, aprendiendo a manejar repositorios remotos a través de Github y simulando un trabajo de proyectos colaborativos.

Al finalizar la sesión, las y los estudiantes podrán:

- Utilizar GitHub como repositorio remoto para publicar y sincronizar proyectos.
- Aplicar un flujo básico de trabajo colaborativo mediante ramas, merges y pull requests.
- Comprender las bases del trabajo en equipo con Git en entornos reales.

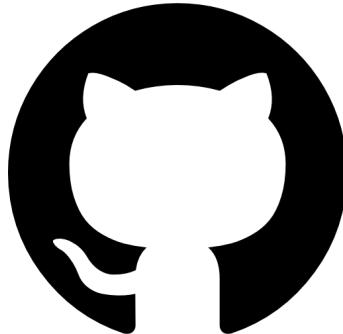
Repaso breve de Git

La clase anterior se estuvo trabajando sobre varios conceptos, recordemos los más esenciales:

Recordamos el flujo local de Git:

- `git init` → Inicialización del repositorio.
- `git add` → Preparación de cambios.
- `git commit` → Registro de versiones.
- `git log` → Consulta del historial.

Rol del repositorio remoto



Para comenzar, hablemos de qué es un repositorio remoto. En la clase 1 trabajamos únicamente de forma local, es decir, con repositorios guardados en nuestras propias computadoras. En esta clase empezaremos a trabajar de manera remota, lo que significa que nuestro repositorio tendrá una copia del proyecto alojada en un **servidor externo, que en este caso será GitHub**.

Un repositorio remoto nos sirve para:

- Respaldar el proyecto.
- Compartir el código y colaborar.
- Hacer visible el avance ante otras personas.
- Sincronizar cambios desde distintos dispositivos.

¿Qué es GitHub?

Es una **plataforma en la nube** que te permite guardar y gestionar proyectos, es donde vamos a guardar nuestro código, alojar repositorios remotos, colaborar y compartir tu código.

GitHub agrega:

- Hosting en la nube
- Herramientas visuales de colaboración
- Solicitudes de cambios (Pull Request),
- registro de tareas o problemas detectados,
- espacios para documentar el proyecto y automatizaciones

Pero el corazón del control de versiones sigue estando en Git mismo, por esta razón, durante este encuentro usaremos Git en local (nuestro equipo)

Comprendemos la relación entre:

- **Repositorio local** → Donde trabajamos.
- **Repositorio remoto** → Donde publicamos los cambios.

Creación de usuario en GitHub

Será necesario contar con una cuenta en Github si no la tenes aun podes crearla sin problema en el siguiente link:

Github: <https://github.com/login>

Sign up for GitHub

Email

Use a different Google account

Username*

Your Country/Region*

For compliance reasons, we're required to collect country information to send you occasional updates and announcements.

Email preferences

Receive occasional product updates and announcements

Create account >

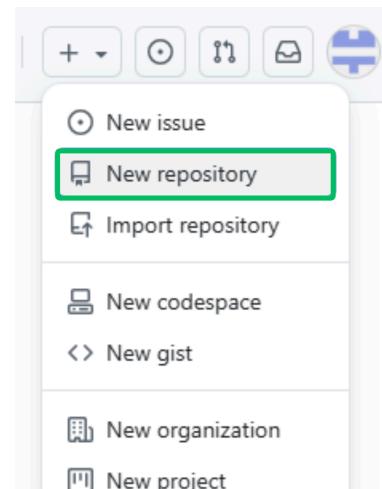
El email y el usuario que crees serán los que utilizaremos para poder comenzar a usar Git

Introducción práctica a GitHub

Ingresamos a la cuenta con la que estuvimos trabajando con Git y creamos un nuevo repositorio remoto.

Pasos iniciales:

1. Iniciamos sesión en GitHub.
2. Creamos un repositorio vacío para este proyecto.
3. Exploramos las secciones principales y que nos serán útiles del repositorio:
 - o **Code**: archivos del proyecto.
 - o **Commits**: historial de versiones.
 - o **Branches**: ramas disponibles.
 - o **Pull Requests**: solicitudes de fusión.



The screenshot shows a GitHub repository named 'Prueba'. The 'Code' tab is active. Below it, there's a summary: '1 Branch' (highlighted by a green box) and '0 Tags'. A search bar says 'Type / to search'. To the right, there are buttons for 'Pin', 'Watch 0', and 'Code'. Below the summary, a commit history is shown: 'talentotech-lab Agregando Subitulo' (commit bab8ea1, yesterday) and 'index.html Agregando Subtitulo' (yesterday). A green box highlights the '3 Commits' link next to the commit details.

Para poder visualizar las secciones será necesario terminar de conectar con el repositorio.

Conectar el repositorio local con GitHub

Es necesario realizar la conexión para que nuestro trabajo que se encontraba de manera local pueda ser guardado en el repositorio remoto.

Comandos fundamentales:

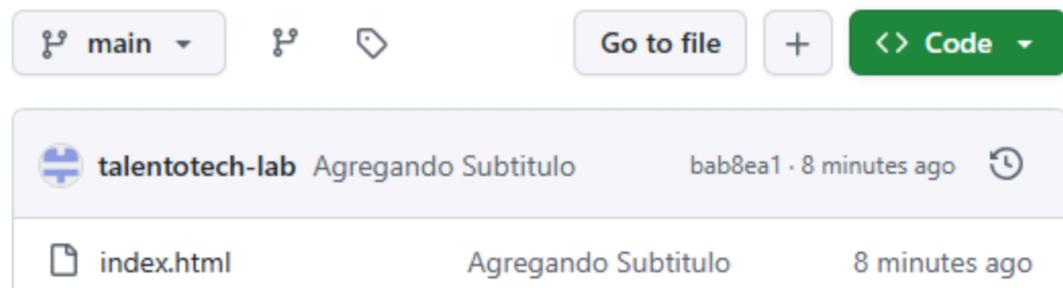
- Agregar el remoto: `git remote add origin <URL>`
- Branch: `git branch -M main` sirve para administrar ramas.
- Primer push: `git push -u origin main` la `-u`: establece a `origin/main` como predeterminado para futuros push y pull.

Ejemplo del código en la siguiente hoja:

```
git remote add origin https://github.com/talentotech-lab/Prueba.git
git branch -M main
git push -u origin main
```

Luego verificamos en GitHub que los archivos y los commits estén correctamente publicados.

Ejemplo de como se verá en github:



Una vez realizamos la conexión de nuestro repositorio local y nuestro repositorio remoto podremos ver todos los archivos, cambios y mensajes que fuimos dejando, que recordemos se los llamó commits.

Mini-práctica 1

Subimos nuestro proyecto a GitHub

Objetivo: conectar el repositorio local con GitHub.

Pasos:

1. Crear repositorio vacío en GitHub.
2. Copiar la URL del repositorio.
3. En la terminal del proyecto ejecutar:
 - o `git remote add origin <URL>`
 - o `git branch -M main` (si es necesario renombrar la rama principal).
 - o `git push -u origin main`
4. Verificar en GitHub que el código esté publicado.

Ramas (branches) y trabajo paralelo

Una **rama** es una línea de trabajo separada que permite desarrollar nuevas ideas sin afectar la versión principal.

Cuándo usar ramas:

- Nuevas funcionalidades.
- Correcciones puntuales.
- Experimentos.

Creando una rama:

- Crear rama: `git branch nombre-rama`
- Cambiar de rama: `git checkout nombre-rama` o `git switch nombre-rama`

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\Talento-tech\Desktop\Proyecto Git> git branch nueva-rama
PS C:\Users\Talento-tech\Desktop\Proyecto Git> git checkout nueva-rama
M     index.html
Switched to branch 'nueva-rama'
PS C:\Users\Talento-tech\Desktop\Proyecto Git>
```

Con estos comandos creaste una rama nueva y cambiaste a esa rama, ahora podés hacer modificaciones y solamente serán afectadas en esa rama y no en la rama principal **main**, podés continuar trabajando y dejando commits de tus avances.

En caso de querer volver a la rama principal deberás escribir el siguiente comando:

- Volver a main: `git checkout main`

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\Talento-tech\Desktop\Proyecto Git> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
```

Fusionar ramas (merge) y conflictos

El proceso de combinar los cambios de dos o más ramas en una sola es conocido como **MERGE**. Se utiliza comúnmente para integrar el trabajo de una rama que se creó para realizar alguna tarea en específico y combinarla en la rama principal (main o master) una vez que la funcionalidad está completa. Esto crea una única secuencia de commits que representa el historial de ambas ramas.

Paso a paso para fusionar:

1. Crear rama.
2. Trabajar y registrar cambios en esa rama.

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\Talento-tech\Desktop\Proyecto Git> git status
On branch nueva-rama
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

no changes added to commit (use "git add" and/or "git commit -a")
PS C:\Users\Talento-tech\Desktop\Proyecto Git> git add .
PS C:\Users\Talento-tech\Desktop\Proyecto Git> git commit -m "Agregando nueva rama"
[nueva-rama 5d5723e] Agregando nueva rama
  1 file changed, 1 insertion(+), 1 deletion(-)
PS C:\Users\Talento-tech\Desktop\Proyecto Git>
```

3. Volver a **main**: Al momento de volver vamos a notar que todo el código o archivos en los que trabajamos desaparecen, pero realmente es que no se encuentran dentro de la rama principal conocida como main.
4. Ejecutar **git merge nombre-rama**: Esto lo hacemos para poder combinar la rama que nombremos con la rama principal, de esta manera lograremos ver todo lo que estuvimos trabajando.

```
PS C:\Users\Talento-tech\Desktop\Proyecto Git> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
PS C:\Users\Talento-tech\Desktop\Proyecto Git> git merge nueva-rama
Updating bab8ea1..5d5723e
Fast-forward
  index.html | 4 +++
  1 file changed, 4 insertions(+)
PS C:\Users\Talento-tech\Desktop\Proyecto Git>
```

Conflictos

Ocurren cuando hay cambios superpuestos en el mismo archivo o línea de código: dos personas editan lo mismo. O cuando una rama edita un archivo que otra rama elimina; Git no sabe cuál versión mantener y requiere intervención manual para resolver la fusión.

- Error que aparece cuando se quiere cambiar de una rama pero no se realizó el commit correspondiente:

```
PS C:\Users\Talento-tech\Desktop\Proyecto Git> git checkout otra-rama
error: Your local changes to the following files would be overwritten by checkout:
      index.html
Please commit your changes or stash them before you switch branches.
Aborting
PS C:\Users\Talento-tech\Desktop\Proyecto Git>
```

- Error al momento de hacer una fusión (merge) cuando el código se sobreescribe:

Consola:

```
PS C:\Users\Talento-tech\Desktop\Proyecto Git> git merge otra-rama
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
Automatic merge failed; fix conflicts and then commit the result.
PS C:\Users\Talento-tech\Desktop\Proyecto Git>
```

Código en VSC:

```
<section>
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
<<<<< HEAD (Current Change)
      <h2>Nueva Rama Agregada</h2>
=====
      <h2>Prueba error</h2>
>>>>> otra-rama (Incoming Change)
      </section>

```

La solución es realizar de manera manual, editando el código desde el editor, para que quede guardado pero no te olvides de escribir la secuencia de pasos de `git add` y `git commit` para guardar esa modificación

```
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
    both modified: index.html

no changes added to commit (use "git add" and/or "git commit -a")
PS C:\Users\Talento-tech\Desktop\Proyecto Git> git add .
PS C:\Users\Talento-tech\Desktop\Proyecto Git> git commit -m "Solucionando error"
[main 980abdd] Solucionando error
PS C:\Users\Talento-tech\Desktop\Proyecto Git>
```

Pull requests y revisión de cambios

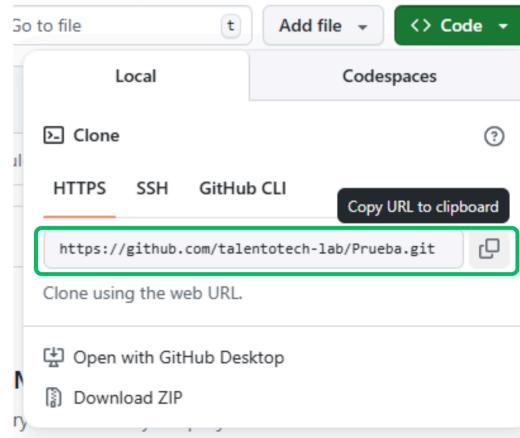
Un **pull request** es una solicitud para fusionar una rama en GitHub. Por lo general se lo utiliza para poder realizar modificación en modo de aporte o propuestas las cuales deberán ser validadas antes de realizar la fusión. Todo esto se realiza desde Github.

Pasos básicos:

1. Ir al repositorio el cual quieras modificar.
2. **Vamos a Fork:** Esto nos permite realizar una copia personal de un repositorio que te permite hacer cambios sin afectar el original.

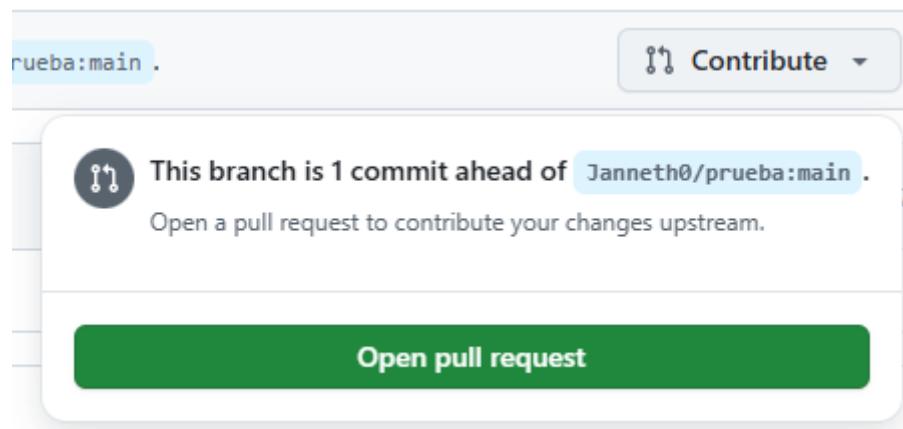


3. Luego realizaremos una clonación del repositorio que queremos modificar escribimos en una nueva terminal y en lo posible una nueva carpeta `git clone <url>` la url la obtenemos de Github.
4. Veremos que todo lo que se estuvo trabajando lo descargamos para poder realizar las modificaciones que necesitemos. **Recordá que el código que estás escribiendo no afectará a la rama principal.**

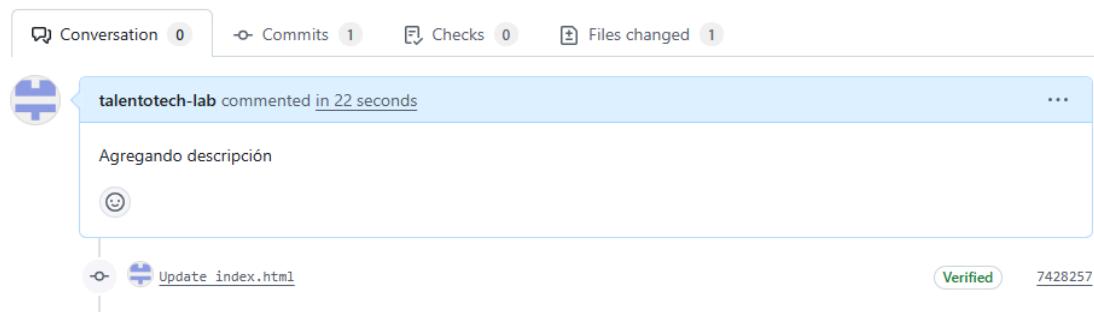


```
PS C:\Users\Talento-tech\Desktop\Proyecto Git> git clone https://github.com/talentotech-lab/Prueba.git
Cloning into 'Prueba'...
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 9 (delta 2), reused 9 (delta 2), pack-reused 0 (from 0)
Receiving objects: 100% (9/9), done.
Resolving deltas: 100% (2/2), done.
PS C:\Users\Talento-tech\Desktop\Proyecto Git>
```

5. Recomendable crear una rama para poder realizar las modificaciones. Una vez realizado los cambios necesarios subir la rama `git push origin nombre-rama`
6. Para poder solicitar revisión debemos ir a nuestro proyecto debes seleccionar Contribute y luego Open pull request.

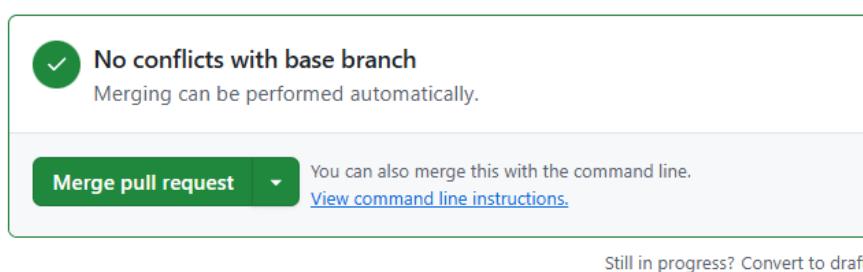


7. Completar título y descripción.



8. Opcional: agregar comentarios o solicitar revisión.

9. Fusionar (merge) desde GitHub.



Diferencia clave:

- **Merge local:** se realiza en tu PC. Con el comando de Git (git merge) fusiona los cambios de una rama a otra en el repositorio local.
- **Pull request:** es una solicitud formal para fusionar código de una rama a otra, un proceso colaborativo que incluye revisión y discusión , se realiza a través de GitHub, ideal para colaboración.

Uso inicial de GitHub Desktop (opcional)

GitHub Desktop permite:

- Clonar repositorios.
- Ver commits y ramas de forma visual.
- Sincronizar cambios con un clic.

Puede resultar útil para quienes recién comienzan y prefieren una interfaz gráfica.

Lo puedes descargar: <https://desktop.github.com/download/>



Cierre conceptual del encuentro

Reconstruimos el flujo completo local → remoto:

1. Trabajo y commits en local.
2. Publicación del código (push).
3. Creación de ramas.
4. Subida de ramas.
5. Inicio y revisión de pull requests.

Este flujo es la base del trabajo colaborativo real y conecta directamente con el Proyecto Integrador.

Mini-práctica 2

Creamos y fusionamos una rama de mejora

Objetivo: practicar creación, modificación y fusión de una rama.

Pasos:

1. Crear rama: `git branch mejora-estilos`
2. Cambiar a la rama: `git checkout mejora-estilos`
3. Realizar mejoras visibles en el código.
4. Registrar cambios mediante commits.
5. Volver a main: `git checkout main`
6. Fusionar: `git merge mejora-estilos`
7. Subir cambios: `git push origin main`

Esta práctica prepara el terreno para el pull request del desafío.

Desafío

Publicamos y colaboramos en un proyecto con Git y GitHub

Objetivo: Aplicar el flujo completo de trabajo con **Git y GitHub**, integrando el versionado local, la publicación en un repositorio remoto y el trabajo colaborativo mediante ramas y pull requests.

Pasos generales:

Conectar el repositorio local con GitHub

- Crear un repositorio remoto en GitHub.
- Vincularlo con tu repositorio local.
- Publicar el proyecto mediante `push`, verificando que los archivos y commits estén visibles en GitHub.

Trabajar con ramas

- Crear una nueva rama para desarrollar una mejora concreta del proyecto.
- Realizar cambios y registrar los avances mediante commits claros y descriptivos.

Simular un flujo colaborativo

- Subir la rama al repositorio remoto.
- Crear un **pull request** desde GitHub para fusionar la rama con **main**.
- Revisar el pull request y completar la fusión.

Este desafío cierra el curso, ya que integra todo lo trabajado en ambas sesiones: flujo local con Git, sincronización con GitHub, trabajo con ramas y revisión de cambios. Comprender este proceso es clave para futuros cursos y para el trabajo en proyectos colaborativos reales.

 **Formato esperado:** El resultado del desafío será un **repositorio en GitHub** que contenga:

- el proyecto versionado iniciado en el **Sesión 1**,
- al menos **una rama creada para una mejora**,
- commits realizados sobre esa rama,
- y un **pull request creado y fusionado** en la rama **main**.

La entrega se realizará mediante **un link de acceso público al repositorio de GitHub**, compartido a través del **campus virtual**, en la tarjeta de entrega ubicada en el mosaico de la **Sesión 2**. Verificá que el repositorio sea **público** o que el enlace permita su correcta visualización

