

Práctico 2: Git y GitHub

Objetivo:

El estudiante desarrollará competencias para trabajar con Git y GitHub, aplicando conceptos fundamentales de control de versiones, colaboración en proyectos y resolución de conflictos, en un entorno simulado y guiado.

Resultados de aprendizaje:

1. Comprender los conceptos básicos de Git y GitHub: Identificar y explicar los principales términos y procesos asociados con Git y GitHub, como repositorios, ramas, commits, forks, etiquetas y repositorios remotos.
2. Manejar comandos esenciales de Git: Ejecutar comandos básicos para crear, modificar, fusionar y gestionar ramas, commits y repositorios, tanto en local como en remoto.
3. Aplicar técnicas de colaboración en GitHub: Configurar y utilizar repositorios remotos, realizar forks, y gestionar pull requests para facilitar el trabajo colaborativo.
4. Resolver conflictos en un entorno de control de versiones: Identificar, analizar y solucionar conflictos de merge generados en un flujo de trabajo con múltiples ramas.

Actividades

- 1) Contestar las siguientes preguntas utilizando las guías y documentación proporcionada (Desarrollar las respuestas) :

- ¿Qué es GitHub?

Es una plataforma que proporciona almacenamiento para repositorios de control de versiones, permitiendo a los desarrolladores guardar y administrar sus proyectos de software facilita la colaboración en equipo y el intercambio de código.

- ¿Cómo crear un repositorio en GitHub?

Para iniciar un repositorio y poder subirlo a GitHub se sigue la siguiente secuencia:

git init: Inicializa un nuevo repositorio de Git en el directorio actual. Esto crea una nueva carpeta oculta llamada `.git` que contiene toda la información de control de versiones del proyecto.

git add . Añade todos los archivos del directorio actual (y sus subdirectorios) al área de preparación. Esto significa que los cambios que realices en esos archivos estarán listos para ser confirmados (commiteados).

git commit -m "nombre del commit": Realiza un commit de los cambios que se encuentran en el área de preparación. El parámetro **-m** permite añadir un mensaje corto que describe los cambios realizados en ese commit.

git branch -m main: Cambia el nombre de la rama actual a "main". El parámetro **-M** es un comando de Git que fuerza el cambio de nombre, incluso si ya existe una rama con el nombre "main".

git remote add origin: Este comando agrega un "remote" a tu repositorio local. El origin es el nombre predeterminado que se le da al repositorio remoto, y la dirección es la URL de tu repositorio en GitHub.

git push origin main (se guarda el repositorio y sus commits en github): Este comando sube tus cambios locales al repositorio remoto. En este caso, se está empujando los cambios de la rama "main" a la rama "main" en el repositorio remoto.

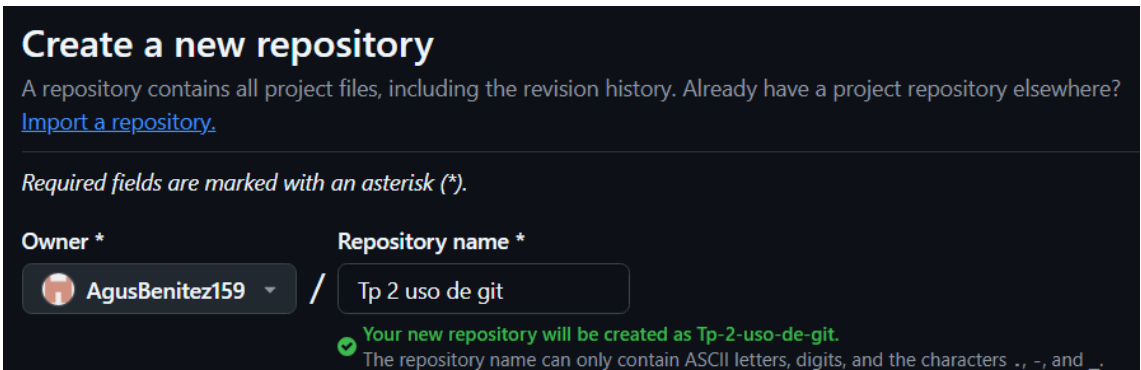
- ¿Cómo crear una rama en Git?
 - `git branch nombre_rama`
- ¿Cómo cambiar a una rama en Git?
 - `git checkout nombre_rama`
- ¿Cómo fusionar ramas en Git?
 - `git merge nombre_rama`
- ¿Cómo crear un commit en Git?
 - `git commit -m "mensaje del commit"`
- ¿Cómo enviar un commit a GitHub?
 - `git push origin nombre_rama`
- ¿Qué es un repositorio remoto?
 - Es una versión de tu repositorio que se encuentra en un servidor (por ejemplo, en GitHub).
- ¿Cómo agregar un repositorio remoto a Git?
 - `git remote add origin <url_del_repositorio>`
- ¿Cómo empujar cambios a un repositorio remoto?
 - `git push origin nombre_rama`
- ¿Cómo tirar de cambios de un repositorio remoto?

- `git pull origin nombre_rama`
- ¿Qué es un fork de repositorio?
 - Es una copia de un repositorio que puedes modificar sin afectar el original.
- ¿Cómo crear un fork de un repositorio?
 - Desde la página de GitHub, haz clic en el botón "Fork" en el repositorio.
- ¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?
 - Desde GitHub, haz clic en "New pull request" y selecciona la rama que quieres fusionar.
- ¿Cómo aceptar una solicitud de extracción?
 - En GitHub, revisa la solicitud y haz clic en "Merge pull request".
- ¿Qué es un etiqueta en Git?
 - Es una marca que se aplica a un punto específico en la historia de un repositorio.
- ¿Cómo crear una etiqueta en Git?
 - `git tag nombre_etiqueta`
- ¿Cómo enviar una etiqueta a GitHub?
 - `git push origin nombre_etiqueta`
- ¿Qué es un historial de Git?
 - Es el registro de todos los commits realizados en el repositorio.
- ¿Cómo ver el historial de Git?
 - `git log`
- ¿Cómo buscar en el historial de Git?
 - `git log --grep="palabra_clave"`
- ¿Cómo borrar el historial de Git?
 - `git reset --hard HEAD~n` (donde "n" es el número de commits a retroceder)
- ¿Qué es un repositorio privado en GitHub?

- Es un repositorio que solo pueden ver las personas a las que se les da acceso explícito.
- ¿Cómo crear un repositorio privado en GitHub?
 - En GitHub, al crear un repositorio, selecciona "Private".
- ¿Cómo invitar a alguien a un repositorio privado en GitHub?
 - En la configuración del repositorio, en "Manage access", invita a un colaborador.
- ¿Qué es un repositorio público en GitHub?
 - Es un repositorio accesible para cualquier persona en Internet.
- ¿Cómo crear un repositorio público en GitHub?
 - En GitHub, al crear un repositorio, selecciona "Public".
- ¿Cómo compartir un repositorio público en GitHub?
 - Comparte la URL del repositorio con otras personas.

2) Realizar la siguiente actividad:

- Crear un repositorio.
 - Dale un nombre al repositorio.
 - Elije el repositorio sea público.
 - Inicializa el repositorio con un archivo.




Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

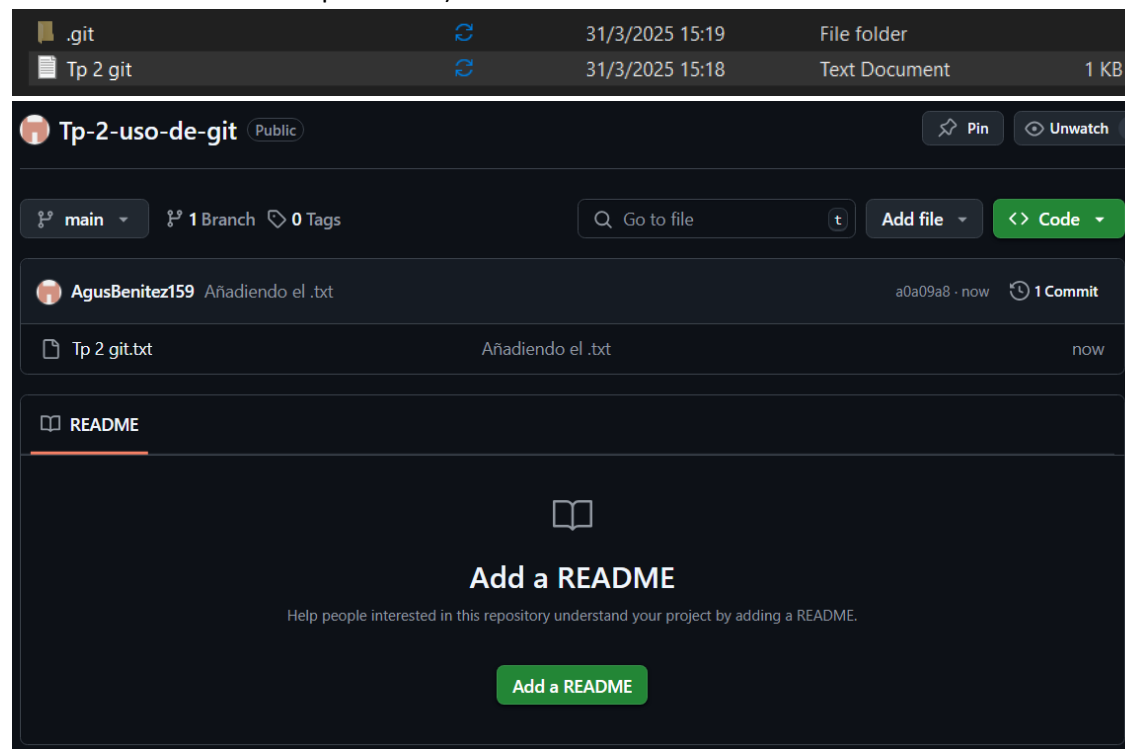
Required fields are marked with an asterisk ().*

Owner * **Repository name ***

 AgusBenitez159 / Tp 2 uso de git

✓ Your new repository will be created as **Tp-2-uso-de-git**.
The repository name can only contain ASCII letters, digits, and the characters `.`, `-`, and `_`.

- Agregando un Archivo
 - Crea un archivo simple, por ejemplo, "mi-archivo.txt".
 - Realiza los comandos `git add .` y `git commit -m "Agregando mi-archivo.txt"` en la línea de comandos.
 - Sube los cambios al repositorio en GitHub con `git push origin main` (o el nombre de la rama correspondiente).



- Creando Branchs
 - Crear una Branch
 - Realizar cambios o agregar un archivo
 - Subir la Branch

```
C:\Users\agust\OneDrive\Escritorio\Tp 2 git>git branch prueba2_
```

3) Realizar la siguiente actividad:

Paso 1: Crear un repositorio en GitHub

- Ve a GitHub e inicia sesión en tu cuenta.
- Haz clic en el botón "New" o "Create repository" para crear un nuevo repositorio.
- Asigna un nombre al repositorio, por ejemplo, conflict-exercise.
- Opcionalmente, añade una descripción.
- Marca la opción "Initialize this repository with a README".
- Haz clic en "Create repository".

Paso 2: Clonar el repositorio a tu máquina local

- Copia la URL del repositorio (usualmente algo como `https://github.com/tuusuario/conflict-exercise.git`).
- Abre la terminal o línea de comandos en tu máquina.
- Clona el repositorio usando el comando:

```
git clone https://github.com/tuusuario/conflict-exercise.git
```

- Entra en el directorio del repositorio: `cd conflict-exercise`

Paso 3: Crear una nueva rama y editar un archivo

- Crea una nueva rama llamada feature-branch:

```
git checkout -b feature-branch
```

- Abre el archivo README.md en un editor de texto y añade una línea nueva, por ejemplo: Este es un cambio en la feature branch.
- Guarda los cambios y haz un commit: `git add README.md` `git commit -m "Added a line in feature-branch"`

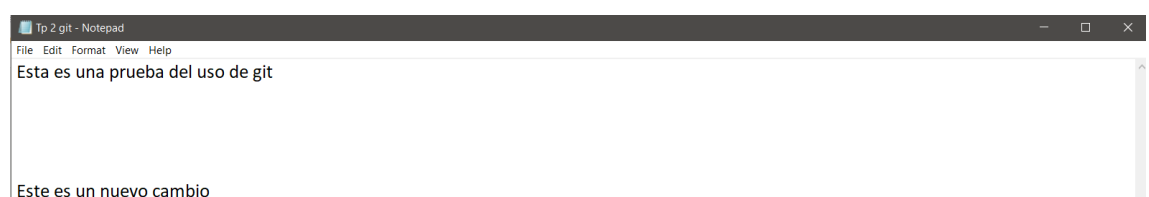
Paso 4: Volver a la rama principal y editar el mismo archivo

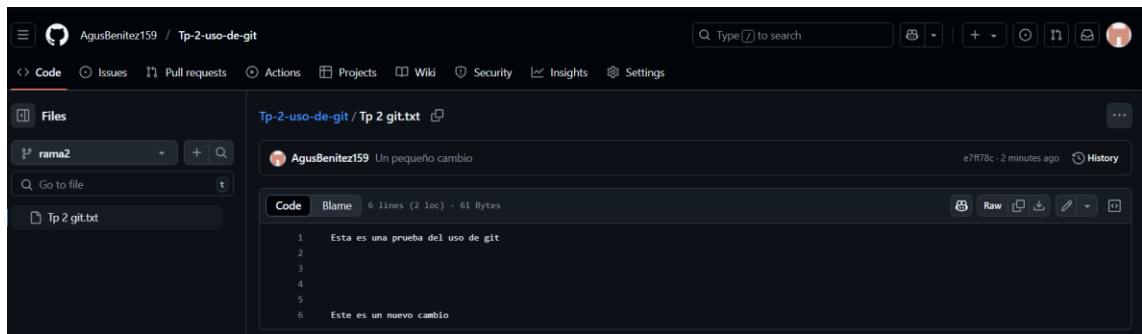
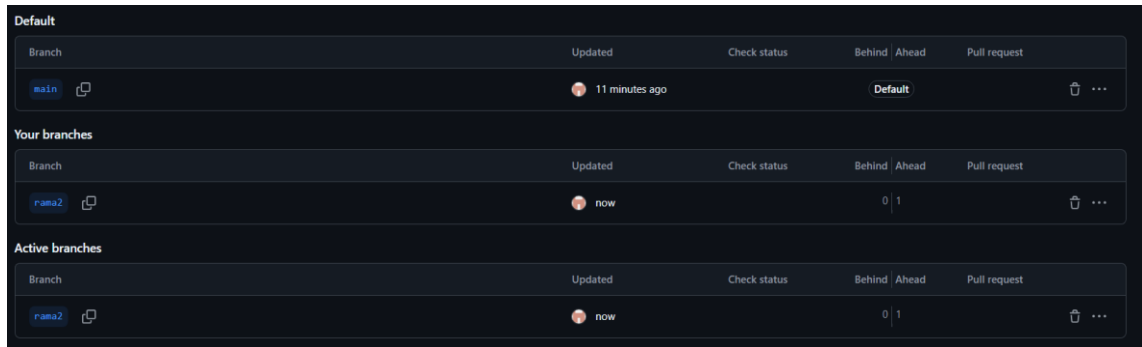
- Cambia de vuelta a la rama principal (main):

```
git checkout main
```

- Edita el archivo README.md de nuevo, añadiendo una línea diferente:

Este es un cambio en la main branch.





- Guarda los cambios y haz un commit: `git add README.md` `git commit -m "Added a line in main branch"`

Paso 5: Hacer un merge y generar un conflicto

- Intenta hacer un merge de la feature-branch en la rama main:

`git merge feature-branch`

- Se generará un conflicto porque ambos cambios afectan la misma línea del archivo README.md.

```
C:\Users\agust\OneDrive\Escritorio\Tp 2 git>git merge rama2
Updating a0a09a8..e7ff78c
Fast-forward
 Tp 2 git.txt | 7 ++++++-
 1 file changed, 6 insertions(+), 1 deletion(-)

C:\Users\agust\OneDrive\Escritorio\Tp 2 git>
```

Paso 6: Resolver el conflicto

- Abre el archivo README.md en tu editor de texto. Verás algo similar a esto:

`<<<<<<< HEAD`

Este es un cambio en la main branch.

```
=====
```

Este es un cambio en la feature branch.

```
>>>>>> feature-branch
```

- Decide cómo resolver el conflicto. Puedes mantener ambos cambios, elegir uno de ellos, o fusionar los contenidos de alguna manera.
- Edita el archivo para resolver el conflicto y guarda los cambios (Se debe borrar lo marcado en verde en el archivo donde estes solucionando el conflicto. Y se debe borrar la parte del texto que no se quiera dejar).
- Añade el archivo resuelto y completa el merge:

```
git add README.md git commit -m
```

```
"Resolved merge conflict"
```

Paso 7: Subir los cambios a GitHub

- Sube los cambios de la rama main al repositorio remoto en GitHub:

```
git push origin main
```

- También sube la feature-branch si deseas:

```
git push origin feature-branch
```

Paso 8: Verificar en GitHub

- Ve a tu repositorio en GitHub y revisa el archivo README.md para confirmar que los cambios se han subido correctamente.
- Puedes revisar el historial de commits para ver el conflicto y su resolución.