

Práctico 2: Git y GitHub

Alumna: Azcurra, Gianina

Comisión 1

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?

GitHub es un portal creado para alojar el código de las aplicaciones de cualquier desarrollador. La plataforma está creada para que los desarrolladores suban el código de sus aplicaciones y herramientas, y que como usuario no solo puedas descargar la aplicación, sino también entrar a su perfil para leer sobre ella o colaborar con su desarrollo.

- ¿Cómo crear un repositorio en GitHub?

Primero debemos crearnos una cuenta en GitHub. Para subir un proyecto a GitHub, se debe crear un repositorio donde alojarlo. En la esquina superior derecha de cualquier página, utilizando el menú desplegable + puedes seleccionar Repositorio Nuevo. Luego debes escribir un nombre corto y fácil de recordar para tu repositorio. Por ejemplo: "hola-mundo". También puedes agregar una descripción de tu repositorio. Por ejemplo, "Mi primer repositorio en GitHub". Luego de eso, también puedes chequear la visibilidad del repositorio y restringir quién

tiene acceso a un repositorio eligiendo la visibilidad del mismo: público o privado.
Para finalizar, tenés que hacer click en Crear repositorio.

- **¿Cómo crear una rama en Git?**

Para crear una rama en Git, primero debes asegurarte de estar dentro de tu repositorio local.

Si no estás en la carpeta correcta, navega hasta ella utilizando el comando `cd` en la terminal. Una vez dentro del repositorio, puedes crear una nueva rama utilizando el comando `git branch` seguido del nombre que desees para la nueva rama, por ejemplo, `git branch feature-branch`.

- **¿Cómo cambiar a una rama en Git?**

Para cambiar a una rama en Git, debes utilizar el comando `git checkout` seguido del nombre de la rama a la que desees cambiarte. Por ejemplo, si quieres cambiarte a una rama llamada `feature-branch`, puedes escribir `git checkout feature-branch`. Esto hará que tu entorno de trabajo se actualice y te permita realizar cambios en esa rama.

- **¿Cómo fusionar ramas en Git?**

Para fusionar ramas en Git, se utiliza el comando `git merge`. El proceso consiste en integrar los cambios de una rama (por lo general, una rama de características o "feature branch") a otra (normalmente, la rama principal como `master` o `main`). Para hacerlo, primero debes asegurarte de estar en la rama en la que desees integrar los cambios. Por ejemplo, si quieres fusionar los cambios de `feature-branch` a la rama principal `main`, debes seguir estos pasos:

Primero, cambia a la rama en la que desees fusionar los cambios, en este caso, `main`, usando el comando `git checkout main`. Después, usando el comando `git merge` seguido del nombre de la rama que desees fusionar (en este caso, `feature-branch`):

- **¿Cómo crear un commit en Git?**

Una vez que se llega a cierto punto en el desarrollo, queremos guardar nuestros cambios (quizás después de una tarea o asunto específico) o subir un archivo / proyecto. `Git commit` es como establecer un punto de control en el proceso de desarrollo al cual puedes volver más tarde si es necesario. También necesitamos escribir un mensaje corto para explicar qué hemos desarrollado o modificado en el código fuente : `git commit -m "mensaje de confirmación"`
Hacemos un commit para guardar nuestro txt y le ponemos un mensaje que explique que hemos hecho.

- **¿Cómo enviar un commit a GitHub?**

Continuando con la respuesta de la pregunta anterior, para enviar el commit a un repositorio remoto en GitHub, se debe usar el comando `git push`.

- **¿Qué es un repositorio remoto?**

Tenemos dos tipos de repositorios local y remoto, ambos son entidades separadas y que a través de los comandos de Git podemos unificar. Los repositorios locales residen en las computadoras de los miembros del equipo. Por el contrario, los repositorios remotos se alojan en un servidor al que pueden acceder todos los miembros del equipo, probablemente en Internet o en una red local.

- **¿Cómo agregar un repositorio remoto a Git?**

Primero deberemos crear un repositorio, sin el archivo `readME`. Una vez que tenemos el archivo agregado y guardado de manera local, tenemos que vincular este repositorio local a un repositorio remoto en GitHub. Para esto vamos a utilizar el comando `git remote add`. Este comando va a tomar el alias nuestro repositorio y la url de nuestro repositorio en GitHub, con esto va a vincularlo con nuestro repositorio local: `git remote add <name> <url>`

El alias que vamos a utilizar para Github es `origin` y para obtener la url de nuestro repositorio, podemos encontrarla al principio de nuestro repositorio:

- **¿Cómo empujar cambios a un repositorio remoto?**

A través del comando `Git Push Origin`, podemos enviar archivos incluidos en el commit al repositorio remoto.

- **¿Cómo tirar de cambios de un repositorio remoto?**

Para obtener los cambios de un repositorio remoto en Git, puedes usar el comando `git pull`. Este comando realiza dos acciones:

1. Hace un `git fetch`, es decir, descarga los cambios del repositorio remoto.
2. Hace un `git merge`, es decir, fusiona esos cambios con tu rama local.

- **¿Qué es un fork de repositorio?**

Un fork de un repositorio es una copia personal de un repositorio que resides en tu propia cuenta en una plataforma de alojamiento de código, como GitHub. Al hacer un fork, obtienes una réplica completa del proyecto original, que incluye su historial de cambios y ramas.

- **¿Cómo crear un fork de un repositorio?**

Para crear un fork de un repositorio, primero debes acceder a la página del repositorio original en una plataforma como GitHub. Una vez allí, en la parte superior derecha de la página,

encontrarás un botón que dice Fork. Al hacer clic en este botón, GitHub creará una copia del repositorio en tu cuenta. Después de crear el fork, puedes clonar esta nueva copia a tu máquina local usando el comando `git clone` con la URL de tu repositorio forkeado. Desde allí, puedes trabajar en los archivos localmente y realizar cambios.

¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?

En el repositorio en GitHub hay un banner que te permite abrir un Pull Request para la rama que acabas de subir. Primero deberías abrir Compare & Pull Request, asegurándote de que la rama base sea la correcta (generalmente main o master del repositorio original) y que la rama de comparación sea la que acabas de subir. Luego Añadís un título y una descripción explicando los cambios que hiciste. Finalmente, se debe hacer click en Create Pull Request.

- **¿Cómo aceptar una solicitud de extracción?**

En la parte superior de la página de tu repositorio hay una pestaña llamada Pull Request que te lleva a una lista de todas las solicitudes, y adonde puedes ver cuáles aceptar.

- **¿Qué es un etiqueta en Git?**

Es una referencia que se utiliza para marcar puntos específicos en la historia del proyecto, generalmente para señalar versiones importantes o hitos de desarrollo, como un lanzamiento (release).

- **¿Cómo crear una etiqueta en Git?**

Para crear una etiqueta en Git, puedes optar por dos tipos principales: ligeras y anotadas. Una etiqueta ligera es simplemente una referencia a un commit específico sin información adicional, y se crea con el comando `git tag nombre-etiqueta`. En cambio, una **etiqueta anotada** incluye más información, como el autor, la fecha y un mensaje, y se crea con el comando `git tag -a nombre-etiqueta -m "Mensaje de la etiqueta"`.

- **¿Cómo enviar una etiqueta a GitHub?**

Para enviar una etiqueta específica a GitHub, utiliza el siguiente comando: `git push origin nombre-etiqueta`. Si tienes varias etiquetas y deseas enviarlas todas a GitHub de una vez, puedes usar el siguiente comando: `git push --tags`.

- **¿Qué es un historial de Git?**

El historial de Git es un registro detallado de todos los cambios realizados en un repositorio a lo largo del tiempo. Este historial incluye información sobre los commits (confirmaciones de cambios), las ramas y las etiquetas asociadas, así como las acciones realizadas por los colaboradores.

- **¿Cómo ver el historial de Git?**

Puedes ver el historial de Git utilizando el comando: `git log`.

- **¿Cómo buscar en el historial de Git?**

Para **buscar en el historial de Git**, puedes usar varios comandos y opciones que te permiten filtrar y localizar commits específicos según ciertos criterios, como el contenido de los cambios, el autor, las fechas, o incluso el mensaje de commit.

1. **Buscar por mensaje de commit:**

```
git log --grep="palabra-clave"
```

2. **Buscar por autor**

```
git log --author="nombre del autor"
```

3. **Buscar por fecha**

```
git log --since="fecha"
```

```
git log --until="fecha"
```

```
git log --since="fecha-inicial" --until="fecha-final"
```

- **¿Cómo borrar el historial de Git?**

Para eliminar los últimos commits, puedes usar el comando `git reset --hard HEAD~n`, donde `n` es el número de commits que deseas borrar. Este comando eliminará los commits y los cambios, por lo que es importante tener cuidado. Si solo deseas eliminar los commits pero mantener los cambios en tu directorio de trabajo, puedes usar `git reset --soft HEAD~n`. Si necesitas borrar commits más antiguos o hacer modificaciones, puedes utilizar el comando `git rebase -i HEAD~n`, lo que te permitirá editar, eliminar o modificar los commits anteriores. Para borrar todo el historial de un repositorio, puedes eliminar la carpeta `.git` y volver a inicializar el repositorio con `git init`, pero ten en cuenta que esto eliminará todo el historial y empezará desde cero.

- **¿Qué es un repositorio privado en GitHub?**

Un repositorio privado en GitHub es un repositorio cuyo acceso está restringido, es decir, solo las personas que tú invites o que tengan permisos específicos pueden ver, editar o contribuir al contenido del repositorio.

- **¿Cómo crear un repositorio privado en GitHub?**

Cuando creas el repositorio, tienes la opción de clickear en público o privado. Es decir, al momento de crear el repo podés decidir como querés que sea.

- **¿Cómo invitar a alguien a un repositorio privado en GitHub?**

Para invitar a alguien a un repositorio privado en GitHub, debes ser el propietario o tener permisos de administrador en dicho repositorio. Desde la ventana settings, accedés a "Manage Access" y luego clickeas en el botón "Invite a collaborator". Aparecerá una ventana donde podrás buscar a la persona que quieres invitar. Puedes buscar por nombre de usuario o correo

electrónico asociado a su cuenta de GitHub. Una vez encuentres al usuario, haz clic en "Add" junto a su nombre. Esto enviará una invitación al usuario.

- **¿Qué es un repositorio público en GitHub?**

Un repositorio público en GitHub es un repositorio cuyo contenido está disponible para cualquier persona en internet. Esto significa que cualquiera puede ver, clonar, bifurcar (fork) o contribuir al proyecto, dependiendo de los permisos establecidos.

- **¿Cómo crear un repositorio público en GitHub?**

Cuando creas el repositorio, tienes la opción de clickear en público o privado. Es decir, al momento de crear el repo podés decidir como querés que sea. En este caso, luego de ingresar un nombre para el repositorio, seleccionas "Public".

- **¿Cómo compartir un repositorio público en GitHub?**

Una vez que tienes la URL del repositorio, podés compartir el enlace, invitar colaboradores desde "Manage Access" y luego "Invite Collaborator" o simplemente desde GitHub a través del botón "share".

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 *
✓ Programacion1_Practica is available.

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

Description (optional)

☒ ☐ Public
Anyone on the internet can see this repository. You choose who can commit.

☐ ☐ Private
You choose who can see and commit to this repository.

Initialize this repository with:


☒ Add a README file
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

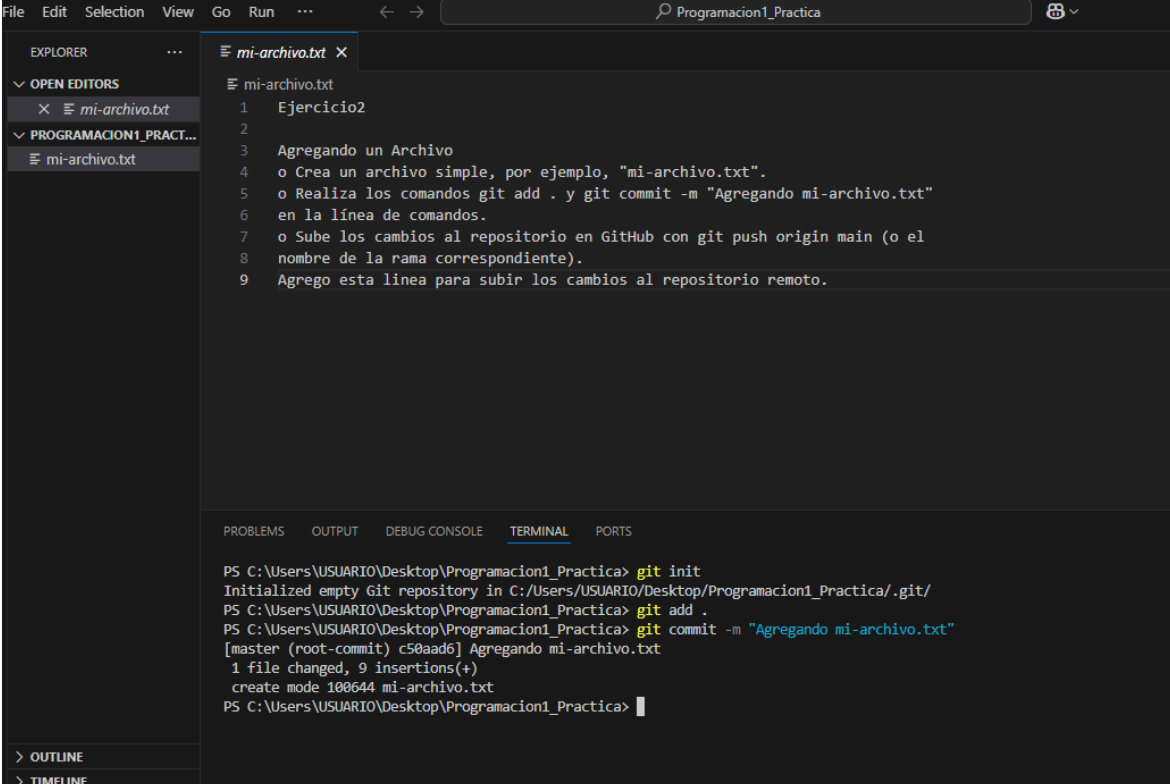
A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set  main as the default branch. Change the default name in your [settings](#).

 You are creating a public repository in your personal account.

Create repository

- 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).

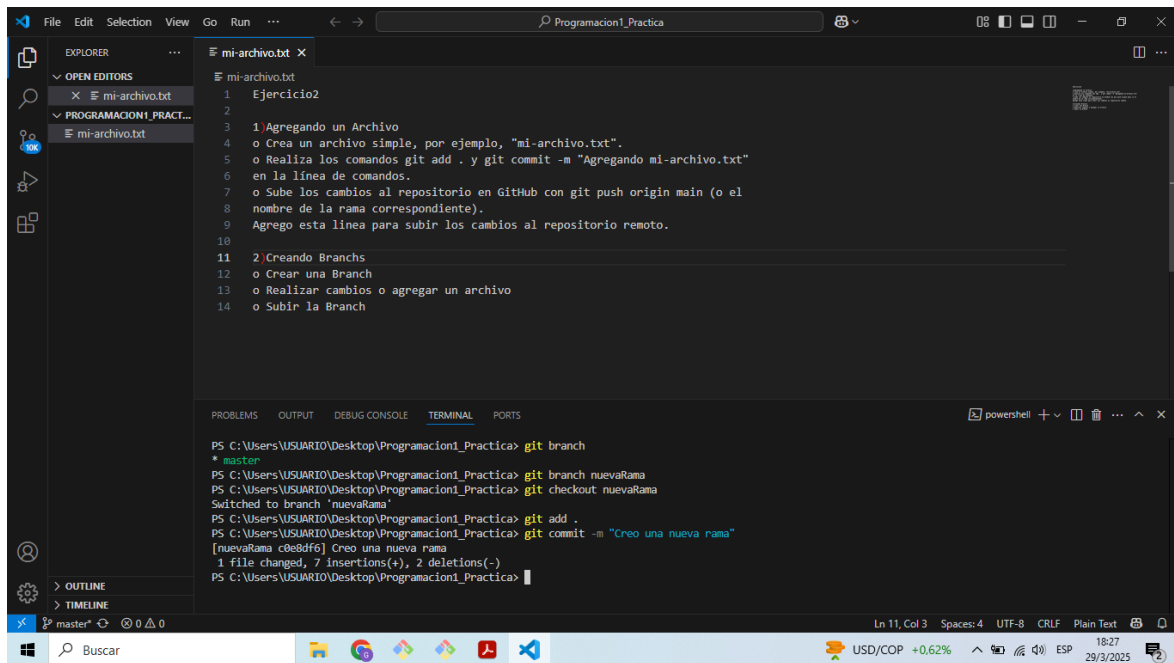


The image shows a screenshot of the Visual Studio Code editor interface. The Explorer panel on the left shows the file structure with 'mi-archivo.txt' selected. The Editor panel displays the content of 'mi-archivo.txt', which includes a title 'Ejercicio2' and instructions for creating a file and committing it to a Git repository. The Terminal panel at the bottom shows the execution of the following commands:

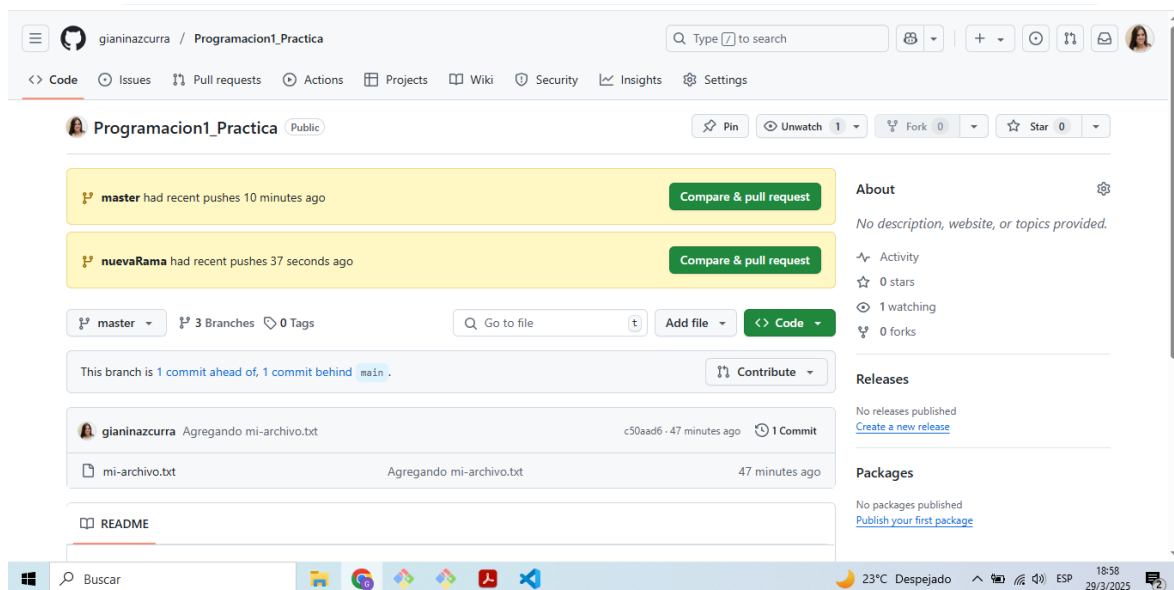
```
PS C:\Users\USUARIO\Desktop\Programacion1_Practica> git init
Initialized empty Git repository in C:/Users/USUARIO/Desktop/Programacion1_Practica/.git/
PS C:\Users\USUARIO\Desktop\Programacion1_Practica> git add .
PS C:\Users\USUARIO\Desktop\Programacion1_Practica> git commit -m "Agregando mi-archivo.txt"
[master (root-commit) c50aad6] Agregando mi-archivo.txt
1 file changed, 9 insertions(+)
create mode 100644 mi-archivo.txt
PS C:\Users\USUARIO\Desktop\Programacion1_Practica>
```

Creando Branchs

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



```
PS C:\Users\USUARIO\Desktop\Programacion1_Practica> git branch
* master
PS C:\Users\USUARIO\Desktop\Programacion1_Practica> git branch nuevaRama
PS C:\Users\USUARIO\Desktop\Programacion1_Practica> git checkout nuevaRama
Switched to branch 'nuevaRama'
PS C:\Users\USUARIO\Desktop\Programacion1_Practica> git add .
PS C:\Users\USUARIO\Desktop\Programacion1_Practica> git commit -m "Creo una nueva rama"
[nuevaRama c8b0dfe] Creo una nueva rama
1 file changed, 7 insertions(+), 2 deletions(-)
PS C:\Users\USUARIO\Desktop\Programacion1_Practica>
```



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".

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 *

conflict-exercise is available.

Great repository names are short and memorable. Need inspiration? How about [fuzzy-octo-journey](#) ?

Description (optional)

☒ ☐ Public
Anyone on the internet can see this repository. You choose who can commit.

☐ ☐ Private
You choose who can see and commit to this repository.








Initialize this repository with:

☒ Add a README file
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

 23°C Despejado

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:

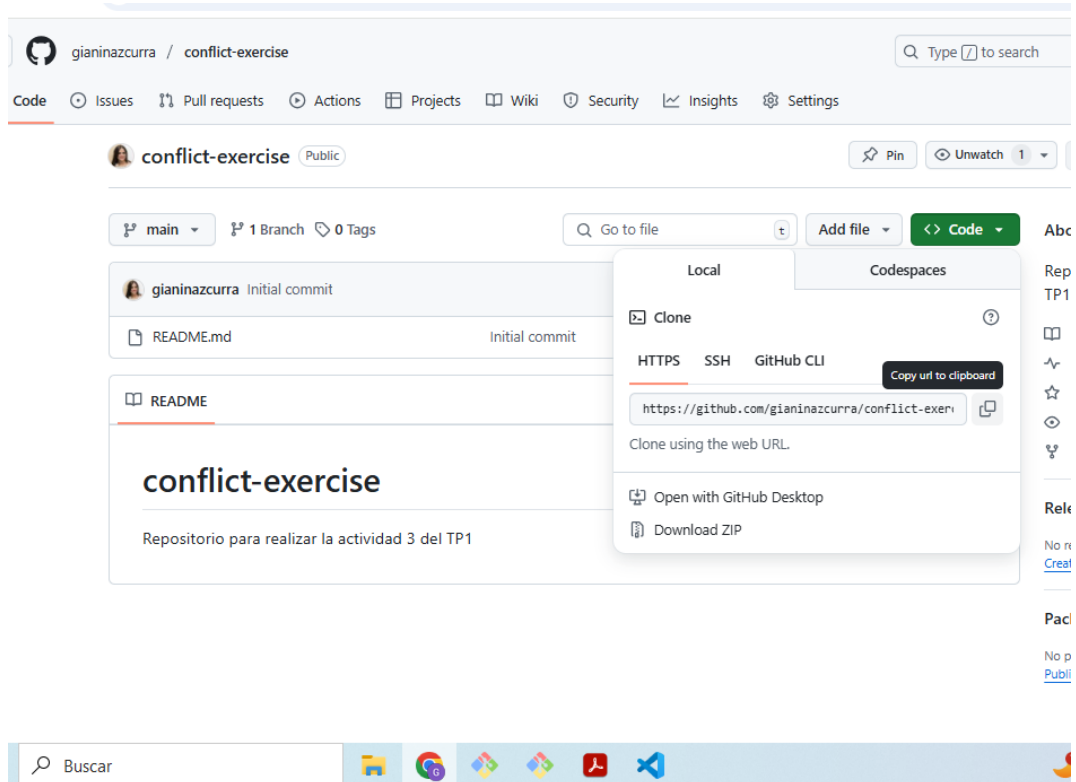
Este es un cambio en la feature branch.

- 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



gianinazcurra / conflict-exercise

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

conflict-exercise Public

main 1 Branch 0 Tags

gianinazcurra Initial commit

README.md Initial commit

README

conflict-exercise

Repositorio para realizar la actividad 3 del TP1

Clone

HTTPS SSH GitHub CLI

Copy url to clipboard

<https://github.com/gianinazcurra/conflict-exercise>

Clone using the web URL

Open with GitHub Desktop

Download ZIP

MINGW64:/c/Users/USUARIO/conflict-exercise

```
USUARIO@DESKTOP-J5O5SDN MINGW64 ~ (master)
$ git clone https://github.com/gianinazcurra/conflict-exercise.git
Cloning into 'conflict-exercise'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.

USUARIO@DESKTOP-J5O5SDN MINGW64 ~ (master)
$ cd conflict-exercise

USUARIO@DESKTOP-J5O5SDN MINGW64 ~/conflict-exercise (main)
$ git checkout -b feature-branch
Switched to a new branch 'feature-branch'

USUARIO@DESKTOP-J5O5SDN MINGW64 ~/conflict-exercise (feature-branch)
$ pwd
/c/Users/USUARIO/conflict-exercise

USUARIO@DESKTOP-J5O5SDN MINGW64 ~/conflict-exercise (feature-branch)
$ git branch
* feature-branch
  main

USUARIO@DESKTOP-J5O5SDN MINGW64 ~/conflict-exercise (feature-branch)
$ git add README.md

USUARIO@DESKTOP-J5O5SDN MINGW64 ~/conflict-exercise (feature-branch)
$ git commit -m "Added a line in feature-branch"
On branch feature-branch
nothing to commit, working tree clean

USUARIO@DESKTOP-J5O5SDN MINGW64 ~/conflict-exercise (feature-branch)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

USUARIO@DESKTOP-J5O5SDN MINGW64 ~/conflict-exercise (main)
$ git add .

USUARIO@DESKTOP-J5O5SDN MINGW64 ~/conflict-exercise (main)
$ git commit -m "Agrego una linea en la rama main"
[main 72375e7] Agrego una linea en la rama main
1 file changed, 1 insertion(+)

USUARIO@DESKTOP-J5O5SDN MINGW64 ~/conflict-exercise (main)
$ git merge feature-branch
Auto-merging README.md
```

```
MINGW64:/c/Users/USUARIO/conflict-exercise
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

USUARIO@DESKTOP-JS0SSDN MINGW64 ~/conflict-exercise (main)
$ git add .

USUARIO@DESKTOP-JS0SSDN MINGW64 ~/conflict-exercise (main)
$ git commit -m "Agrego una linea en la rama main"
[main 72375e7] Agrego una linea en la rama main
1 file changed, 1 insertion(+)

USUARIO@DESKTOP-JS0SSDN MINGW64 ~/conflict-exercise (main)
$ git merge feature-branch
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.

USUARIO@DESKTOP-JS0SSDN MINGW64 ~/conflict-exercise (main|MERGING)
$ git add README.md

USUARIO@DESKTOP-JS0SSDN MINGW64 ~/conflict-exercise (main|MERGING)
$ git commit -m "Resolved merge conflict"
[main 1836421] Resolved merge conflict

USUARIO@DESKTOP-JS0SSDN MINGW64 ~/conflict-exercise (main)
$ git push origin main
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 4 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (9/9), 824 bytes | 164.00 KiB/s, done.
Total 9 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (3/3), done.
To https://github.com/gianinazcurra/conflict-exercise.git
10dd895..1836421 main -> main

USUARIO@DESKTOP-JS0SSDN MINGW64 ~/conflict-exercise (main)
$ git push origin feature-branch
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'feature-branch' on GitHub by visiting:
remote:   https://github.com/gianinazcurra/conflict-exercise/pull/new/feature
-branch
remote:
To https://github.com/gianinazcurra/conflict-exercise.git
* [new branch]   feature-branch -> feature-branch

USUARIO@DESKTOP-JS0SSDN MINGW64 ~/conflict-exercise (main)
$ |
```

- 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.

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:

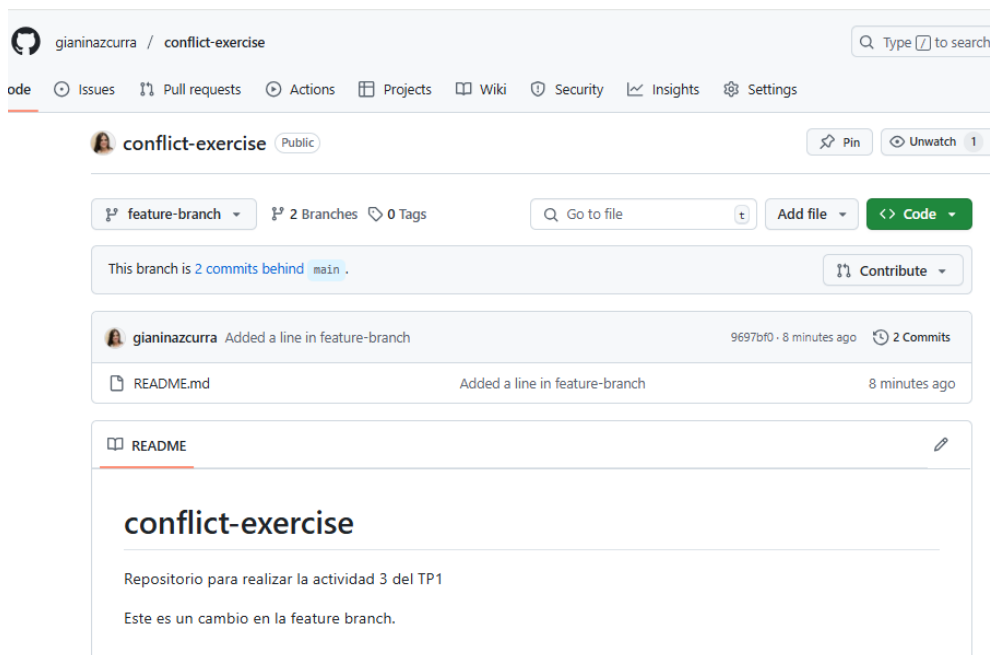
```
git push origin main
```

- 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.



gianinazcurra / conflict-exercise

feature-branch 2 Branches 0 Tags

This branch is 2 commits behind main.

gianinazcurra Added a line in feature-branch 9697bf0 · 8 minutes ago 2 Commits

README.md Added a line in feature-branch 8 minutes ago

README

conflict-exercise

Repositorio para realizar la actividad 3 del TP1

Este es un cambio en la feature branch.



gianinazcurra / conflict-exercise

main 2 Branches 0 Tags

gianinazcurra Resolved merge conflict 1836421 · 1 minute ago 4 Commits

README.md Resolved merge conflict 1 minute ago

README

conflict-exercise

Repositorio para realizar la actividad 3 del TP1

<<<<<< HEAD Este es un cambio en la main branch.