

## Trabajo Práctico N°2

### Git y Github

#### Alumna

Lucía Chamorro

#### Comisión

11

**Tecnicatura Universitaria en Programación - Universidad Tecnológica Nacional.**

**Programación I**

## **Introducción**

### **Objetivos**

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?
- ¿Cómo crear un repositorio en GitHub?
- ¿Cómo crear una rama en Git?
- ¿Cómo cambiar a una rama en Git?
- ¿Cómo fusionar ramas en Git?
- ¿Cómo crear un commit en Git?
- ¿Cómo enviar un commit a GitHub?
- ¿Qué es un repositorio remoto?
- ¿Cómo agregar un repositorio remoto a Git?
- ¿Cómo empujar cambios a un repositorio remoto?
- ¿Cómo tirar de cambios de un repositorio remoto?
- ¿Qué es un fork de repositorio?
- ¿Cómo crear un fork de un repositorio?

- ¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?
- ¿Cómo aceptar una solicitud de extracción?
- ¿Qué es una etiqueta en Git?
- ¿Cómo crear una etiqueta en Git?
- ¿Cómo enviar una etiqueta a GitHub?
- ¿Qué es un historial de Git?
- ¿Cómo ver el historial de Git?
- ¿Cómo buscar en el historial de Git?
- ¿Cómo borrar el historial de Git?
- ¿Qué es un repositorio privado en GitHub?
- ¿Cómo crear un repositorio privado en GitHub?
- ¿Cómo invitar a alguien a un repositorio privado en GitHub?
- ¿Qué es un repositorio público en GitHub?
- ¿Cómo crear un repositorio público en GitHub?
- ¿Cómo compartir un repositorio público en GitHub?

## 2) Realizar la siguiente actividad:

- **Crear un repositorio.**

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

- **Agregando un Archivo**

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

- **Creando Branchs**

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

### 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.

#### 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.



## Actividad 1

### ¿Qué es GitHub?

GitHub es una plataforma en línea que permite almacenar proyectos que usan el sistema de control de versiones Git. Es el servicio más popular del mundo para compartir y colaborar en proyectos de software, tanto de código abierto como privados. Almacena los repositorios en la nube, lo que facilita el trabajo colaborativo entre desarrolladores ubicados en diferentes partes del mundo. Además de servir como un "espacio remoto" donde subir tu código, GitHub incluye herramientas para organizar, discutir, revisar y aprobar cambios, lo que lo convierte en una solución integral para el desarrollo de software moderno.

GitHub también ofrece acciones automatizadas (GitHub Actions), integración con herramientas externas, estadísticas de contribución y perfiles colaborativos.

### ¿Cómo crear un repositorio en GitHub?

Se debe ingresar en el sitio de GitHub [github.com](https://github.com), iniciar sesión o registrarse. Luego ir a a **créate New repository**. Se elige un nombre, se puede ingresar una descripción y elegir si va a ser público o privado. Si es público cualquiera que ingrese a internet puede acceder al repositorio si es privado solo los que se les envíe invitación.

Luego, se vincula el proyecto local con el GitHub, se copia la URL de del repositorio creado y desde la carpeta del proyecto: `git remote add origin` (y se agrega el link), esto le dice a Git que "origin" será el repositorio remoto. Después se envía el trabajo local a GitHub.

Si la rama se llama master, es buena práctica renombrarla a main: `git branch -M main`. Se suben los cambios a GitHub: `git push -u origin main`.

Y, por último, se confirma en GitHub, ingresando al repositorio en GitHub y actualizando la página se verán los archivos subidos

¿Cómo crear una rama en Git?

Se crea una nueva rama con `git Branch` (más el nombre de la nueva rama)

¿Cómo cambiar a una rama en Git?

Se cambia con `git checkout` (nombre de la rama)

¿Cómo fusionar ramas en Git?

Con `git merge` (nombre rama): Fusiona una rama con la actual. para hacer un merge correctamente, se debe estar posicionado en la rama donde se van a integrar los cambios. En general el main, `git checkout main`, se verifica el estado actual del repositorio: `git status`, se trae los cambios desde la rama: `git merge` (nombre rama) y se confirma que el historial refleje la fusión: `git log --oneline --graph`

¿Cómo crear un commit en Git?

Se utiliza `git commit -m` (Se agrega la descripción)

¿Cómo enviar un commit a GitHub?

Se utiliza `git push origin main`, en caso de que sea de la rama principal, si es de otra rama se cambia el nombre

¿Qué es un repositorio remoto?

Son plataformas que permiten sincronizar el trabajo con otros desarrolladores, por ej: GitHub.

¿Cómo agregar un repositorio remoto a Git?

Con `git clone (URL)`, se crea una copia local de un repositorio remoto.

¿Cómo empujar cambios a un repositorio remoto?

Con `git push`, se suben cambios al repositorio remoto

¿Cómo tirar de cambios de un repositorio remoto?

Con `git pull` se descarga los cambios desde el repositorio remoto y los fusiona con la versión local.

¿Qué es un fork de repositorio?

Un fork es una copia completa de un repositorio de GitHub que se crea dentro de la cuenta.

¿Cómo crear un fork de un repositorio?

Se ingresa a un repositorio público de GitHub. Luego se hace clic en el botón Fork (esquina superior derecha) . Y, por último, se elige la cuenta para crear una copia del proyecto en tu

propio GitHub.

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

Primero se debe realizar un fork y clonarlo con `git clone (url)`, Luego entrar al directorio: `cd proyecto` Una vez hecho los cambios y se hayan confirmado (`git add + git commit`), se pueden subir con: `git push origin main` Y luego enviar un pull request desde GitHub para que los responsables del proyecto original revisen y, si lo desean, integren los cambios.

¿Cómo aceptar una solicitud de extracción?

Se evalúa y se consulta si es necesario y se acepta la merge pull request

¿Qué es una etiqueta en Git?

Las etiquetas o tags son marcadores de algún punto específico del repositorio que resulte importante. El uso más común es para marcar los lanzamientos de versiones

¿Cómo crear una etiqueta en Git?

Se crean con `git tag (nombre)`

¿Cómo enviar una etiqueta a GitHub?

Se utiliza `git push origin (nombre de la etiqueta)`

¿Qué es un historial de Git?

Es un registro de todos los cambios que se han realizado en el repositorio. Cada modificación

queda registrada junto con el autor, la fecha y un mensaje explicativo. Esto permite auditar los cambios, rastrear errores o incluso deshacer modificaciones si es necesario.

¿Cómo ver el historial de Git?

Se utiliza `git log` para ver el historial

¿Cómo buscar en el historial de Git?

Con `git log --oneline` se puede ver en una sola línea, obtener una vista rápida, también si se busca algo más específico por ejemplo por autor `git log --author=(Nombre del autor)`

¿Cómo borrar el historial de Git?

Hay tres modos que realizan modificaciones: Modo soft (suave) `git reset --soft`: Retrocede el puntero HEAD al commit especificado, pero no modifica el Stage. Modo mixed (mixto) `git reset --mixed`: Retrocede el puntero HEAD al commit especificado incluyendo también el Stage, y Modo hard (duro) `git reset --hard`: Igual que mixed, pero también modifica de forma permanente los archivos y carpetas del proyecto.

¿Qué es un repositorio privado en GitHub?

Es un repositorio que solo puede acceder la persona que lo creó y a quienes este invite. No es accesible a cualquier persona en internet como lo es un repositorio público.

### ¿Cómo crear un repositorio privado en GitHub?

Se debe ingresar a la plataforma de GitHub con una cuenta y en configuración elegir la opción private

### ¿Cómo invitar a alguien a un repositorio privado en GitHub?

En la sección de configuración ir a collaborators y luego add people, se puede hacer la invitación a través del nombre de usuario git o email.

### ¿Qué es un repositorio público en GitHub?

Es un repositorio al cual puede acceder cualquier persona que navegue en internet. Allí podrá ver y acceder a los proyectos que se hayan subido (archivos, documentación, código, etc)

### ¿Cómo crear un repositorio público en GitHub?

Al igual que con el repositorio privado hay que ir a configuración en GitHub y en este caso se debe elegir la opción public

### ¿Cómo compartir un repositorio público en GitHub?

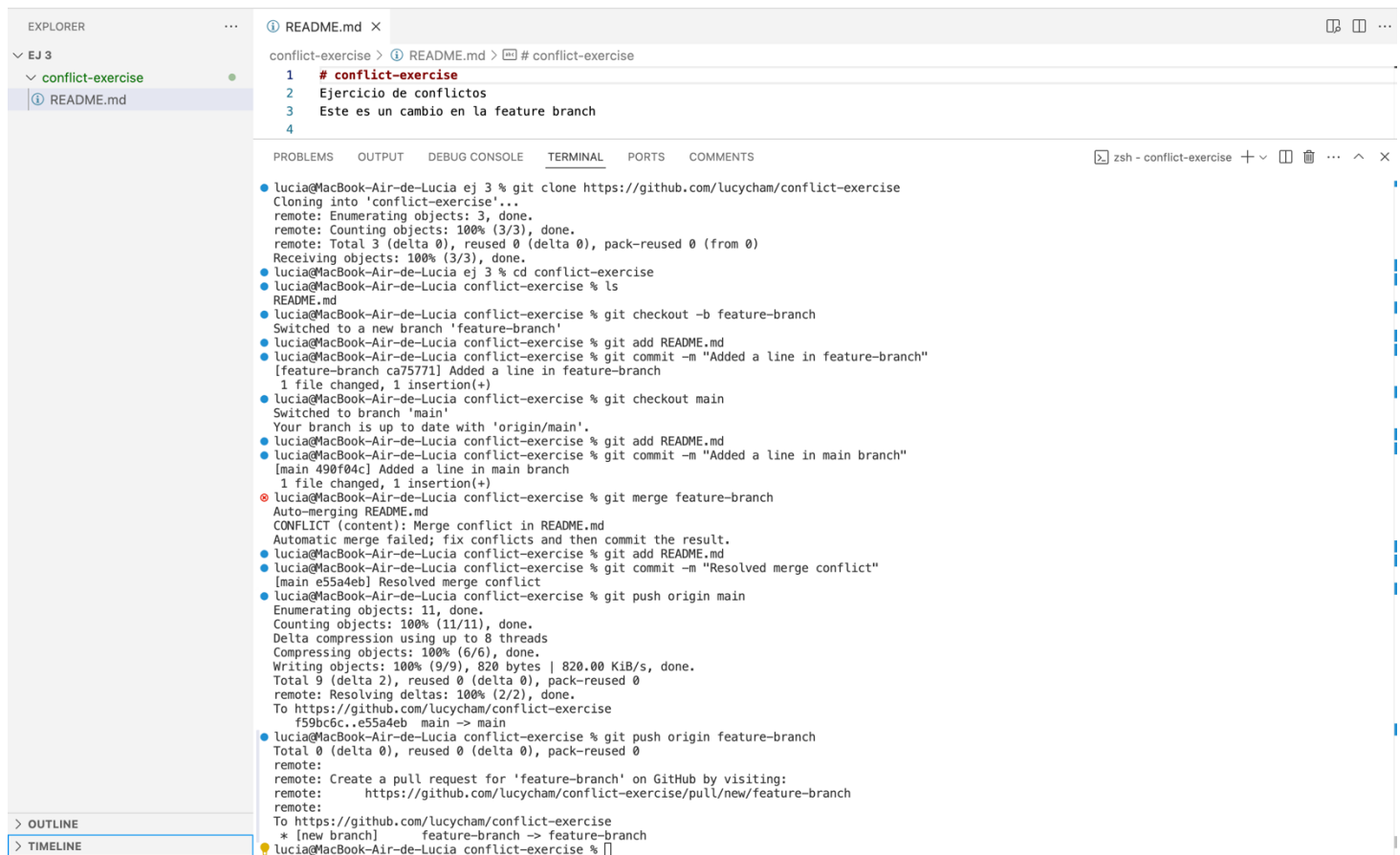
Se puede compartir el link de la url desde el repositorio en GitHub copiándolo de la barra de direcciones.

## Actividad 2

Link <https://github.com/lucycham/primer-repo-git>

## Actividad 3

Link <https://github.com/lucycham/conflict-exercise>



The screenshot shows a VS Code editor with a file explorer on the left showing a project named 'conflict-exercise' with a 'README.md' file. The main editor area displays the content of 'README.md':

```
1 # conflict-exercise
2 Ejercicio de conflictos
3 Este es un cambio en la feature branch
4
```

Below the editor, the 'TERMINAL' tab is active, showing a series of git commands and their output:

```
Lucia@MacBook-Air-de-Lucia ej 3 % git clone https://github.com/lucycham/conflict-exercise
Cloning into 'conflict-exercise'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
Lucia@MacBook-Air-de-Lucia ej 3 % cd conflict-exercise
Lucia@MacBook-Air-de-Lucia conflict-exercise % ls
README.md
Lucia@MacBook-Air-de-Lucia conflict-exercise % git checkout -b feature-branch
Switched to a new branch 'feature-branch'
Lucia@MacBook-Air-de-Lucia conflict-exercise % git add README.md
Lucia@MacBook-Air-de-Lucia conflict-exercise % git commit -m "Added a line in feature-branch"
[feature-branch ca75771] Added a line in feature-branch
1 file changed, 1 insertion(+)
Lucia@MacBook-Air-de-Lucia conflict-exercise % git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
Lucia@MacBook-Air-de-Lucia conflict-exercise % git add README.md
Lucia@MacBook-Air-de-Lucia conflict-exercise % git commit -m "Added a line in main branch"
[main 490f04c] Added a line in main branch
1 file changed, 1 insertion(+)
Lucia@MacBook-Air-de-Lucia conflict-exercise % 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.
Lucia@MacBook-Air-de-Lucia conflict-exercise % git add README.md
Lucia@MacBook-Air-de-Lucia conflict-exercise % git commit -m "Resolved merge conflict"
[main e55a4eb] Resolved merge conflict
Lucia@MacBook-Air-de-Lucia conflict-exercise % git push origin main
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (9/9), 820 bytes | 820.00 KiB/s, done.
Total 9 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/lucycham/conflict-exercise
f59bc6c..e55a4eb main -> main
Lucia@MacBook-Air-de-Lucia conflict-exercise % git push origin feature-branch
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'feature-branch' on GitHub by visiting:
remote:   https://github.com/lucycham/conflict-exercise/pull/new/feature-branch
remote:
To https://github.com/lucycham/conflict-exercise
* [new branch]   feature-branch -> feature-branch
Lucia@MacBook-Air-de-Lucia conflict-exercise %
```