Trabajo Práctico Nº 2: Git y GitHub

1) **Contestar las siguientes preguntas utilizando las guías y documentación proporcionada:**

• ¿Qué es GitHub?
GitHub es una plataforma donde nosotros podemos compartir nuestros repositorios de forma pública y/o privada. A su vez podremos acceder a otros repositorios de otras personas.

• ¿Cómo crear un repositorio en GitHub?
Una vez creado un perfil en esa plataforma, voy a “Repositories” y hago click donde dice “New” (es un botón verde). Allí deberé asignarle un nombre al repositorio, una descripción, definir si es público o privado y finalmente darle click al botón “Create repository”.

• ¿Cómo crear una rama en Git?
Por defecto yo inicializo mi repositorio en la rama “main”. Desde allí puedo crear ramas para distintas features a trabajar, es por ello que desde “main” ejecuto el comando:
git branch <nombre_de_la_rama>
Luego puedo chequear las ramas que tengo disponibles con el comando:
git branch
Allí veremos donde estamos parados y la rama que creamos.

• ¿Cómo cambiar a una rama en Git?
Para cambiarnos a la rama que creamos ejecutamos el comando:
git checkout <nombre_de_la_rama>

• ¿Cómo fusionar ramas en Git?
Una vez que trabajamos la feature que necesitábamos en esa rama, debemos incorporar esas modificaciones al código principal que tenemos en la rama main. Para ello debemos, primero pararnos en la rama a la que queremos mergear:
git checkout main
En este caso sería main. Luego, ya parados en la rama a la que queremos mergear ejecutamos el código que permite el “mergeo”:
git merge <nombre_de_la_rama>
En caso de haber algún conflicto, lo corregimos en función de cómo queremos que quede definitivamente el código en la branch y una vez hecho esto ejecutamos:
git add <nombre_del_archivo_corregido>
git commit -m “Conflicto xxxxx corregido”
Y allí quedará ejecutado el merge.

• ¿Cómo crear un commit en Git?
Un “commit” es un recurso que tenemos para “sacarle una foto” a nuestro código y dejar asentadas las diferencias con su versión anterior. Para ello debemos subir los cambios a un “stage” ejecutando primero:

git add <archivo/os_a_commitear>
Y luego ejecutar el commit que registra los cambios con:
git commit -m "Descripción del commit"

• ¿Cómo enviar un commit a GitHub?
Para enviar un commit a GitHub debemos primero verificar que esté seteado en mi repo local con el siguiente comando:
git remote -v
Si eso esta correcto ejecutamos el comando:
git push origin main

• ¿Qué es un repositorio remoto?
Es un repositorio de Git almacenado en un servidor o servicio en línea como ser GitHub, GitLab, etc. que nos permite compartir y colaborar con otros desarrolladores.

• ¿Cómo agregar un repositorio remoto a Git?
En caso de que luego de ejecutar "*git remote -v"* no nos devuelve nada la consola, entonces debemos agregar el repositorio remoto a nuestro git. Esto se hace con el comando:
git remote add origin <url_del_repositorio>

• ¿Cómo empujar cambios a un repositorio remoto?
Luego de hacer el commit del último cambio que quisimos registrar ejecutamos el siguiente comando:
git push origin main

• ¿Cómo tirar de cambios de un repositorio remoto?
Hay veces que el repositorio remoto sufre modificaciones que no se reflejaron en mi repositorio local, para ello debemos "tirar" los cambios del repositorio remoto con el comando:
git pull origin main

• ¿Qué es un fork de repositorio?
En caso de que queramos tener una copia completa de un repositorio de otro desarrollador podemos "forkearlo" lo cual crea un repositorio exactamente igual en nuestro Github.

• ¿Cómo crear un fork de un repositorio?
En el repositorio remoto hay un botón que tiene la leyenda "Fork" haciendo click allí nos abre un formulario para editar el nombre como lo vamos a llamar en nuestro Github y luego poder copiarlo.

• ¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?
En el repositorio forkeado se debe hacer click en el botón "Compare & Pull Request", selecciono la branch a pullear y también debo prestar atención de elegir la rama original con la que quiero hacer el merge. Agrego una descripción explicando las modificaciones y luego hago click en "Create pull request".

• ¿Cómo aceptar una solicitud de extracción?
En principio se debe tener permisos de colaborador o ser dueño del repositorio. Luego debemos revisar los cambios y aprobarlos o rechazarlos según corresponda. También podemos realizar comentarios sin rechazar o aceptar. Si consideramos que la PR está en condiciones de ser integrada al proyecto hacemos click en “Confirm Merge”.

• ¿Qué es una etiqueta en Git?
Una etiqueta, también conocida con “Tag”, es una referencia que señala un punto específico en la historia de un proyecto. Se usan para marcar versiones o hitos importantes (ej. “v1.0”, “v2.5”, etc). A diferencia de las ramas, estas son fijas y no se pueden cambiar una vez creada.

• ¿Cómo crear una etiqueta en Git?
Se pueden crear de forma liviana, la cual se usa solo para marcar un punto en la historia del proyecto:
git tag <nombre_de_la_etiqueta>
Pero también existen las anotadas, las cuales tienen info del autor, fecha y un mensaje. El comando es:
git tag -a <nombre_de_la_etiqueta> -m “Mensaje descriptivo”

• ¿Cómo enviar una etiqueta a GitHub?
Para enviar la etiqueta ejecutamos el comando:
git push origin <nombre_de_la_etiqueta>
o si queremos enviar todas las etiquetas de una vez:
git push –tags

• ¿Qué es un historial de Git?
Es el registro de todos los commits del repositorio.

• ¿Cómo ver el historial de Git?
Para verlos debemos ejecutar el comando:
git log

• ¿Cómo buscar en el historial de Git?
Para buscar en el historial del repositorio tenemos algunas alternativas. Por ejemplo podemos buscar por palabras o frases claves en los mensajes del commit con el siguiente comando:
git log --grep="palabra clave"
También podemos buscar por autor:
git log --author="Gabriel Gudiño"
Por fechas:
git log --since="2025-01-01"
git log --since="2025-01-01" --until="2025-01-31"
Y varias alternativas más.

• ¿Cómo borrar el historial de Git?
Para borrar los últimos commits, pero manteniendo los archivos modificados ejecuto:
git reset –soft HEAD~1
Siendo 1 el valor de los últimos commits que queremos borrar.
Si estos commits ya habían sido pusheados ejecutamos el comando siguiente para borrarlos en el repositorio remoto:
git push –force

• ¿Qué es un repositorio privado en GitHub?
Es un repositorio que está restringido a un grupo selecto de colaboradores.

• ¿Cómo crear un repositorio privado en GitHub?
Luego de iniciar sesión, voy a “Repositories”, hago click en “New” y se me abre un formulario de creación de un repositorio. Luego del nombre y la descripción del repositorio tengo la opción de elegir Publico o Privado.

• ¿Cómo invitar a alguien a un repositorio privado en GitHub?
En el mismo repositorio hay una pestaña llamada “Settings” y dentro de ella buscamos la opción de “Collaborators” donde podremos invitar colaboradores por su username, email o Nombre Completo.

• ¿Qué es un repositorio público en GitHub?
Es un repositorio al que cualquier persona en internet puede ver, clonar y/o descargar sin necesidad de permisos especiales.

• ¿Cómo crear un repositorio público en GitHub?
Luego de iniciar sesión, voy a “Repositories”, hago click en “New” y se me abre un formulario de creación de un repositorio. Luego del nombre y la descripción del repositorio tengo la opción de elegir Publico o Privado.

• ¿Cómo compartir un repositorio público en GitHub?
Al abrir el repositorio en la pagina de Github hay un botón verde con la leyenda “<> Code”. Al hacer click alli no aparece una dirección HTTPS del repositorio para poder compartir.

2) **Realizar la siguiente actividad:**

• Crear un repositorio.
    - o Dale un nombre al repositorio.
    - o Elige 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.

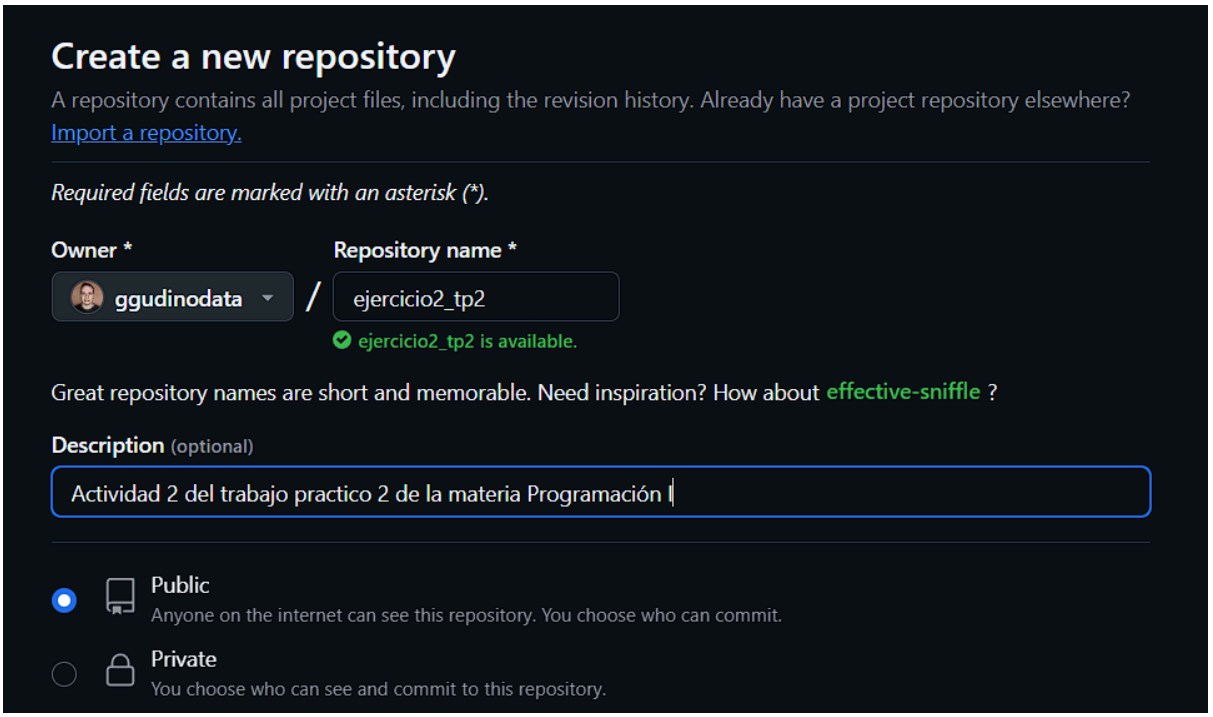
o Sube los cambios al repositorio en GitHub con git push origin main (o el nombre de la rama correspondiente).

• Creando Branchs

o Crear una Branch

o Realizar cambios o agregar un archivo

o Subir la Branch




```
ggudino@ggudino:~/repositories/ejercicio2_tp2$ git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /home/ggudino/repositories/ejercicio2_tp2/.git/
```

```
ggudino@ggudino:~/repositories/ejercicio2_tp2$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        mi-archivo.txt

nothing added to commit but untracked files present (use "git add" to track)
```

```
ggudino@ggudino:~/repositories/ejercicio2_tp2$ git add .
ggudino@ggudino:~/repositories/ejercicio2_tp2$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   mi-archivo.txt
```

```
ggudino@ggudino:~/repositories/ejercicio2_tp2$ git commit -m "first commit"
[master (root-commit) 692c814] first commit
 1 file changed, 1 insertion(+)
 create mode 100644 mi-archivo.txt
```

```
ggudino@ggudino:~/repositories/ejercicio2_tp2$ git push -u origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 257 bytes | 257.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/ggudinodata/ejercicio2_tp2.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

```
ggudino@ggudino:~/repositories/ejercicio2_tp2$ git checkout -b main
Switched to a new branch 'main'
ggudino@ggudino:~/repositories/ejercicio2_tp2$ git status
On branch main
nothing to commit, working tree clean
ggudino@ggudino:~/repositories/ejercicio2_tp2$ git branch
* main
  master
```

```
ggudino@ggudino:~/repositories/ejercicio2_tp2$ git checkout -b main
Switched to a new branch 'main'
ggudino@ggudino:~/repositories/ejercicio2_tp2$ git status
On branch main
nothing to commit, working tree clean
ggudino@ggudino:~/repositories/ejercicio2_tp2$ git branch
* main
  master
ggudino@ggudino:~/repositories/ejercicio2_tp2$ git status
On branch main
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:   mi-archivo.txt

no changes added to commit (use "git add" and/or "git commit -a")
ggudino@ggudino:~/repositories/ejercicio2_tp2$ git add .
ggudino@ggudino:~/repositories/ejercicio2_tp2$ git commit -m "Cambios realizados al archivo de prueba"
[main b5fb1ab] Cambios realizados al archivo de prueba
 1 file changed, 2 insertions(+), 1 deletion(-)
```

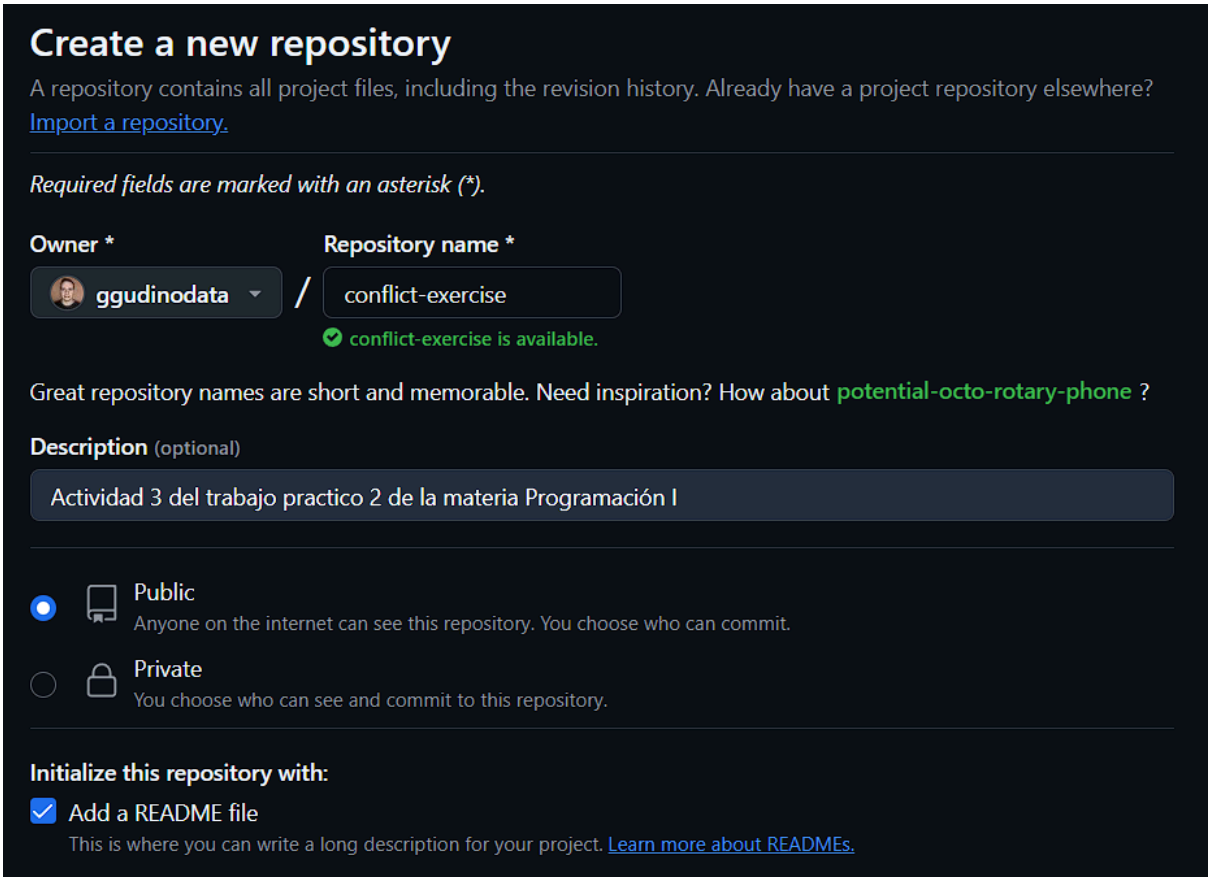
```
ggudino@ggudino:~/repositories/ejercicio2_tp2$ git push -u origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 317 bytes | 317.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'main' on GitHub by visiting:
remote:      https://github.com/ggudinodata/ejercicio2_tp2/pull/new/main
remote:
To https://github.com/ggudinodata/ejercicio2_tp2.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

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.

Paso 1:



Paso 2:

```
ggudino@ggudino:~/repositories/conflict-exercise$ git remote -v
origin  https://github.com/ggudinodata/conflict-exercise.git (fetch)
origin  https://github.com/ggudinodata/conflict-exercise.git (push)
```

Paso 3:

```
ggudino@ggudino:~/repositories/conflict-exercise$ git checkout -b feature-branch
Switched to a new branch 'feature-branch'
```

```
ggudino@ggudino:~/repositories/conflict-exercise$ git diff
diff --git a/README.md b/README.md
index 32be856..14a2217 100644
--- a/README.md
+++ b/README.md
@@ -1,2 +1,3 @@
 # conflict-exercise
 Actividad 3 del trabajo practico 2 de la materia Programación I
+Este es un cambio en la feature branch.
```

```
ggudino@ggudino:~/repositories/conflict-exercise$ git commit -m "Added a line in feature-branch"
[feature-branch 91185e7] Added a line in feature-branch
 1 file changed, 1 insertion(+)
```

Paso 4:

```
ggudino@ggudino:~/repositories/conflict-exercise$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
ggudino@ggudino:~/repositories/conflict-exercise$ git diff
diff --git a/README.md b/README.md
index 32be856..e43f000 100644
--- a/README.md
+++ b/README.md
@@ -1,2 +1,3 @@
 # conflict-exercise
 Actividad 3 del trabajo practico 2 de la materia Programación I
+Añadimos una linea diferente
\ No newline at end of file
```

```
ggudino@ggudino:~/repositories/conflict-exercise$ git add README.md
ggudino@ggudino:~/repositories/conflict-exercise$ git commit -m "Added a line in main branch"
[main bf0a08c] Added a line in main branch
 1 file changed, 1 insertion(+)
```

Paso 5:

```
ggudino@ggudino:~/repositories/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.
```

README.md ! ×

README.md > # Actividad 3 del trabajo practico 2 de la materia Programación I<<<<<<< HEADAñadimos una linea diferente

```
1  # conflict-exercise
2  Actividad 3 del trabajo practico 2 de la materia Programación I
   Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
3  <<<<<<< HEAD (Current Change)
4  Añadimos una linea diferente
5  =======
6  Este es un cambio en la feature branch.
7  >>>>>>> feature-branch (Incoming Change)
8
```

Paso 6:

README.md ! ●

README.md > # conflict-exercise

```
1  # conflict-exercise
2  Actividad 3 del trabajo practico 2 de la materia Programación I
3  Este es un cambio en la feature branch.
4  Añadimos una linea diferente.
```

```
ggudino@ggudino:~/repositories/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.
ggudino@ggudino:~/repositories/conflict-exercise$ git add README.md
ggudino@ggudino:~/repositories/conflict-exercise$ git commit -m "Resolved merge conflict"
[main 4c5e52a] Resolved merge conflict
```

Paso 7:

```
ggudino@ggudino:~/repositories/conflict-exercise$ git push origin main
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 16 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (9/9), 831 bytes | 831.00 KiB/s, done.
Total 9 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), done.
To https://github.com/ggudinodata/conflict-exercise.git
   e4cfc32..4c5e52a  main -> main
ggudino@ggudino:~/repositories/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/ggudinodata/conflict-exercise/pull/new/feature-branch
remote:
To https://github.com/ggudinodata/conflict-exercise.git
 * [new branch]      feature-branch -> feature-branch
```

Paso 8: