

Práctico 2: Git y GitHub

Alumna: Melina Yangüez

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:

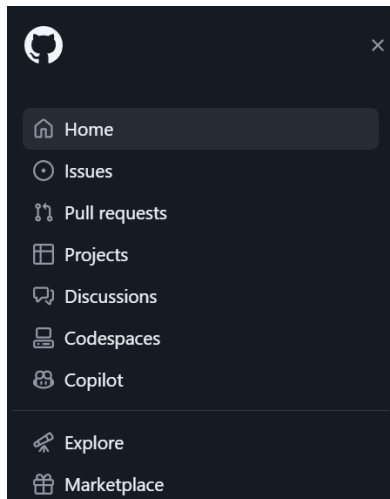
1. Comprender los conceptos básicos de Git y GitHub: Identificar y explicar los principales términos y procesos asociados con Git y GitHub, como repositorios, ramas, commits, forks, etiquetas y repositorios remotos.
2. Manejar comandos esenciales de Git: Ejecutar comandos básicos para crear, modificar, 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

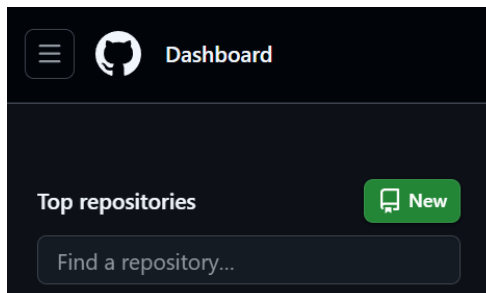
- 1) Contestar las siguientes preguntas utilizando las guías y documentación proporcionada (Desarrollar las respuestas) :
 - ¿Qué es GitHub?

Es una plataforma que ofrece alojamiento de repositorios de control de versiones, que permite a los desarrolladores almacenar y gestionar sus proyectos de software. Es una herramienta gratuita y de código abierto que permite el trabajo en equipo en proyectos, el intercambio de código y el trabajo conjunto de forma eficiente.
 - ¿Cómo crear un repositorio en GitHub?

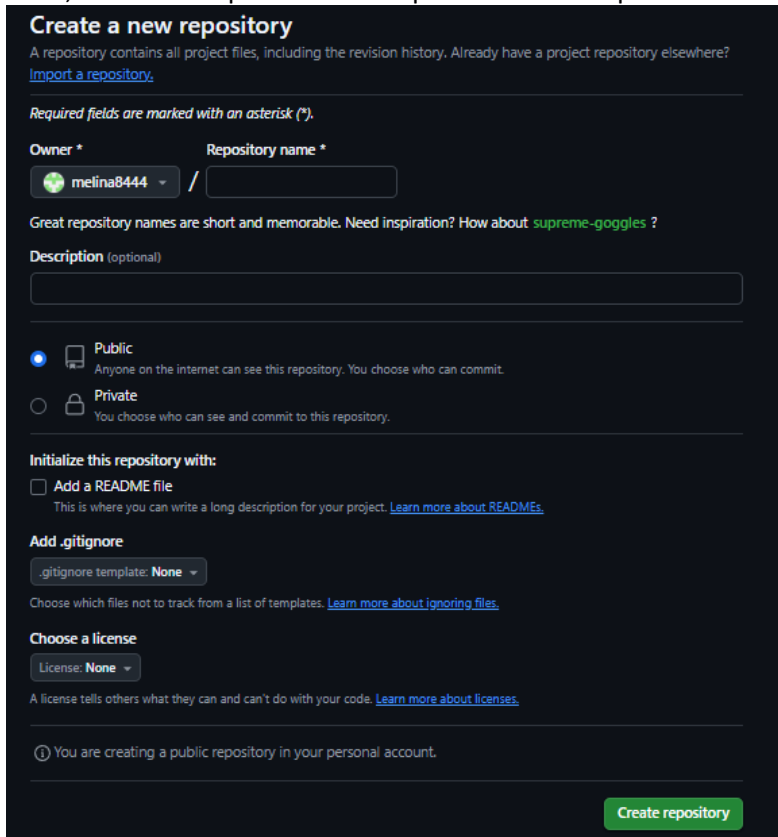
Para empezar a utilizar GitHub, primero debes crear una cuenta. Puedes hacerlo en la página web de GitHub. Una vez que tengas una cuenta, puedes configurar tu perfil y empezar a crear repositorios.
- a) Si es la primera vez utilizando esta plataforma y queremos desde nuestro git subir un repositorio a github debemos ir al menú, seleccionar Home.



Luego, clickear en New:



Una vez allí debemos colocar el nombre, alguna que otra descripción de lo que vamos a subir, si queremos que sea público o privado, etc.

A screenshot of the 'Create a new repository' form on GitHub. The form has a title 'Create a new repository' and a subtitle 'A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)'. Below this, there's a note 'Required fields are marked with an asterisk (*)'. The form fields are: 'Owner *' (a dropdown menu showing 'melina8444'), 'Repository name *' (a text input field), 'Description (optional)' (a text input field), 'Public' (selected radio button) and 'Private' (unselected radio button) options, 'Initialize this repository with:' section with 'Add a README file' (checkbox) and a note 'This is where you can write a long description for your project. [Learn more about READMEs.](#)', 'Add .gitignore' section with a dropdown menu showing '.gitignore template: None', 'Choose a license' section with a dropdown menu showing 'License: None', and a note 'A license tells others what they can and can't do with your code. [Learn more about licenses.](#)'. At the bottom, there's a note 'You are creating a public repository in your personal account.' and a green 'Create repository' button.

Una vez que esta todo, damos click en Create Repository.

Nos saldrán varias líneas de comandos como:

...or create a new repository on the comand line: Nos indica cada uno de los pasos que debemos de hacer para poder mandar desde nuestro repositorio local al repositorio que hemos creado en github y enlazarlos (recomendado cuando empiezas desde 0).

...or push an existing repository from the command line: Este caso hace referencia a que ya tienes un repositorio local y solamente deseas mandarlo a la plataforma. Para ello deberás de tipear los siguientes comandos en el bash:

```
git remote add origin https://github.com/tucanal/nombre-repo.git
```

```
git push -u origin main
```

b) Clonar un repositorio:

- Click en el repositorio que queremos.
- En el botón verde que dice “Code” hacemos click y copiamos la ulr.
- En la carpeta deseada, colocamos: `git clone https://github.com/tucanal/repositorio.git`
- Para subir los cambios desde un repositorio clonado, realiza `git add` . luego `commit` y haz `git push origin main`

- ¿Cómo crear una rama en Git?

- Escribe en la consola el comando: `git branch nuevaRama`
- El HEAD es un apuntador que indica en que rama estas. Si ejecutas `git branch`, la rama actual aparecerá resaltada.

- ¿Cómo cambiar a una rama en Git?

- `Git checkout nuevaRama`
 - Para crear una nueva rama y saltar a ella, en un solo paso, puedes utilizar el comando `git checkout` con la opción `-b`: `git checkout -b nuevaRama`

- ¿Cómo fusionar ramas en Git?

- Posicionarse en la rama destino (ej: master o main): `git checkout main`
- Fusionar la rama deseada(ej: ramaNueva) en la actual: `git merge nuevaRama`, esto integra los cambios de nuevaRama en main. Si no hay conflictos, git aplicara los cambios automáticamente.

- ¿Cómo crear un commit en Git?

- Realizar los cambios que necesitamos en nuestro archivo.
- Agregamos los cambios al staging área(preparación):

- git add archivo1 (esto es para agregar un archivo específico)
 - git add . (Para agregar todos los archivos modificados)
- Guardar los cambios con un commit (y un mensaje descriptivo):
git commit -m "Mensaje que explica los cambios"
- ¿Cómo enviar un commit a GitHub?
- Para enviar los commits al repositorio en GitHub, debemos empujarlos con: git push origin nombre_de_la_rama
- ¿Qué es un repositorio remoto?
- Para poder colaborar en cualquier proyecto Git, necesitas saber cómo gestionar repositorios remotos. Los repositorios remotos son versiones de tu proyecto que están hospedadas en Internet o en cualquier otra red. Puedes tener varios de ellos, y en cada uno tendrás generalmente permisos de solo lectura o de lectura y escritura. Colaborar con otras personas implica gestionar estos repositorios remotos enviando y trayendo datos de ellos cada vez que necesites compartir tu trabajo. Gestionar repositorios remotos incluye saber cómo añadir un repositorio remoto, eliminar los remotos que ya no son válidos, gestionar varias ramas remotas, definir si deben rastrearse o no y más.
- ¿Cómo agregar un repositorio remoto a Git?
- Vincular un repositorio remoto utilizando: git remote add [nombre][url]

Ej: git remote add origin https://github.com/usuario/repo.git
- Luego verificar repositorios remotos configurados: git remote -v (esto muestra la lista de remotos y sus urls)
- Descargar cambios del remoto (sin fusionar) git fetch origin (esto descarga los cambios del remoto pero no aplica a tu rama local.)
- ¿Cómo empujar cambios a un repositorio remoto?
- Antes de empujar tus cambios, es una buena práctica obtener los últimos cambios del repositorio remoto para evitar conflictos: git pull origin nombre_de_la_rama
- Empuja tus cambios al repositorio remoto: git push origin nombre_de_la_rama
- ¿Cómo tirar de cambios de un repositorio remoto?
- Como mencione en el punto anterior, para tirar los cambios del repositorio remoto Usa el comando: git pull , para descargar y fusionar los cambios del repositorio remoto con tu rama local: git pull origin nombre_de_la_rama
- Por ejemplo, si estás trabajando en la rama main: git pull origin main
- ¿Qué es un fork de repositorio?
- En muchas ocasiones podemos ver en github repositorios que son de nuestro agrado y donde nosotros queremos tener una copia y hacerle algunas modificaciones, para ello github nos ofrece la implementación de fork.
- Buscamos algún repositorio que nos parezca interesante. Este puede ser cualquier proyecto de código abierto que quieras modificar o utilizar como base para tu propio trabajo.

- ¿Cómo crear un fork de un repositorio?
 - Buscar un repositorio publico que te interese en github.
 - Hacer click en “Fork”(esquina superior derecha) para crear una copia en tu cuenta. (el fork será independiente , los cambios que se realicen no afectara al repositorio original)
 - Clonar el fork localmente para trabajar:

Git clone <https://github.com/tu-usuario/repositorio-forkeado.git>

- Una vez realizado, se puede modificar el código en la maquina y subir los cambios solo a nuestra copia.
- Cómo enviar una solicitud de extracción (pull request) a un repositorio?
 - Para hacer el pull request nos dirigiremos a la solapa de Pull requests allí daremos click en new pull request, veremos una ventana a modo de resumen en donde se reflejarán los cambios que hemos hecho nosotros en comparación al repositorio original (el código original, mejor dicho). Daremos click en Create pull request donde veremos el asunto (colocamos algún mensaje global) y más abajo tenemos suficiente lugar para poder explayarnos en mencionar el porque ese cambio que hemos realizado nosotros, sería considerado como algo que a el repositorio original le vendrían bien agregarlo.
- ¿Cómo aceptar una solicitud de extracción?
 - El autor del repositorio verá en sus pull requests el mensaje que le hemos enviado, para que lo pueda observar y si lo considera realizar el cambio pertinente (además de poder responderle al usuario que le ha propuesto ese cambio). Lo bueno de todo esto es que si el usuario original considera que esta modificación es buena y no genera conflictos con la rama maestra de su repositorio local remoto, puede clickear en Merge pull request y de esta manera sumará a su repositorio los cambios que hizo un usuario (en modo de ayuda).
- ¿Qué es un etiqueta en Git?
 - Como muchos VCS, Git tiene la posibilidad de etiquetar puntos específicos del historial como importantes. Esta funcionalidad se usa típicamente para marcar versiones de lanzamiento (v1.0, por ejemplo).
- ¿Cómo crear una etiqueta en Git?
 - Git utiliza dos tipos principales de etiquetas: ligeras y anotadas.
 - Una etiqueta ligera es muy parecida a una rama que no cambia - simplemente es un puntero a un commit específico.
 - Las etiquetas anotadas se guardan en la base de datos de Git como objetos enteros. Tienen un checksum; contienen el nombre del etiquetador, correo electrónico y fecha; tienen un mensaje asociado; y pueden ser firmadas y verificadas con GNU Privacy Guard (GPG). Normalmente se recomienda que crees etiquetas anotadas, de manera que tengas toda esta información; pero si quieres una etiqueta temporal o por alguna razón no estás interesado en esa información, entonces puedes usar las etiquetas ligeras.

- ¿Cómo enviar una etiqueta a GitHub?
 - Primero, debes crear una etiqueta en tu repositorio local.
 - Puedes crear una etiqueta anotada (que incluye información adicional como el autor y la fecha) o una etiqueta ligera (simplemente un puntero a un commit): Ej. etiqueta ligera: `git tag v1.0`
 - Puedes verificar las etiquetas que has creado localmente con: `git tag`
 - Una vez que has creado la etiqueta en tu repositorio local, necesitas empujarla al repositorio remoto en GitHub. Puedes hacer esto con el siguiente comando: `git push origin v1.0`
 - (origin es el nombre del repositorio remoto (por defecto suele ser origin) y v1.0 es el nombre de la etiqueta.)
 - Para empujar todas las etiquetas creadas, usar: `git push origin --tags`
- ¿Qué es un historial de Git?
 - El historial de Git es una secuencia de todos los cambios realizados en un repositorio de Git. Cada cambio en el repositorio se guarda como un commit, y cada commit contiene información sobre el estado del proyecto en un momento específico, incluyendo:
 - Identificador del commit
 - Autor
 - Fecha de realización
 - Mensaje enviado
- ¿Cómo ver el historial de Git?
 - Se consigue con el comando: `git log` (historial completo), estará invertido, es decir en los últimos realizados aparecerán los primeros.
 - `git log --graph --oneline --all` (historial gráfico de las ramas y merges)
 - Para ver un número de logs determinado introducimos ese número como opción, con el signo "-" delante (-1, -8, -12...). Por ejemplo, esto muestra los últimos tres commits: `git log -3`
 - Si queremos que el log también nos muestre los cambios en el código de cada commit podemos usar la opción -p. Esta opción genera una salida mucho más larga, por lo que seguramente nos tocará movernos en la salida con los cursores y usaremos CTRL + Z para salir. `git log -2 -p`
- ¿Cómo buscar en el historial de Git?
 - Para buscar en el historial de commits de Git, puedes utilizar varios comandos y opciones que te permiten filtrar y localizar commits específicos.
 - Para buscar commits que contengan una palabra o frase específica en el mensaje de commit, usa `git log` con la opción `--grep`: `git log --grep="palabra clave"`

- Para buscar commits que han modificado un archivo específico, usa git log seguido del nombre del archivo: `git log -- nombre_del_archivo`
- Para buscar commits en un rango de fechas específico, usa las opciones `--since` y `--until`:
`git log --since="2024-01-01" --until="2024-01-31"`
- Para encontrar commits hechos por un autor específico, usa `--author`: `git log --author="Nombre del Autor"`

- ¿Cómo borrar el historial de Git?

- El comando `git reset` se utiliza sobre todo para deshacer las cosas, como posiblemente puedes deducir por el verbo.
- Se mueve alrededor del puntero HEAD y opcionalmente cambia el index o área de ensayo y también puede cambiar opcionalmente el directorio de trabajo si se utiliza `-hard`.
- Esta última opción hace posible que este comando pueda perder tu trabajo si se usa incorrectamente, por lo que asegúrese de entenderlo antes de usarlo.
- Existen distintas formas de utilizarlo:
 - `git reset` -> Quita del stage todos los archivos y carpetas del proyecto.
 - `git reset nombreArchivo` -> Quita del stage el archivo indicado.
 - `git reset nombreCarpeta/` -> Quita del stage todos los archivos de esa carpeta.
 - `git reset nombreCarpeta/nombreArchivo` -> Quita ese archivo del stage (que a la vez está dentro de una carpeta).
 - `git reset nombreCarpeta/*.extensión` -> Quita todos los archivos que cumplan con la condición indicada previamente dentro de esa carpeta del stage.

- ¿Qué es un repositorio privado en GitHub?

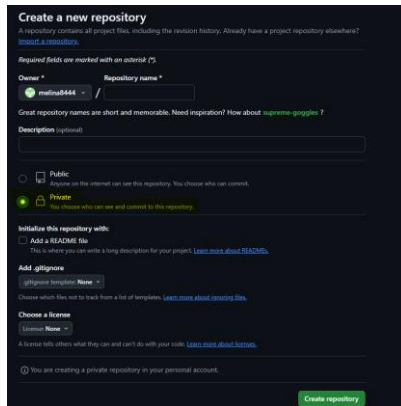
Un repositorio privado en GitHub es un tipo de repositorio en el que el contenido solo es accesible para usuarios específicos que han sido autorizados. A diferencia de los repositorios públicos, donde cualquier persona puede ver y clonar el contenido, un repositorio privado limita el acceso a los colaboradores que tú elijas. Esto es útil para proyectos que contienen información sensible o que aún están en desarrollo y no deseas que estén disponibles públicamente.

- ¿Cómo crear un repositorio privado en GitHub?

- Iniciar sesión: Se debe crear una cuenta en github.com y acceder a la cuenta.
- Crear un repositorio: En la esquina superior derecha, en el signo +, hacer click, luego seleccionar "New Repository".

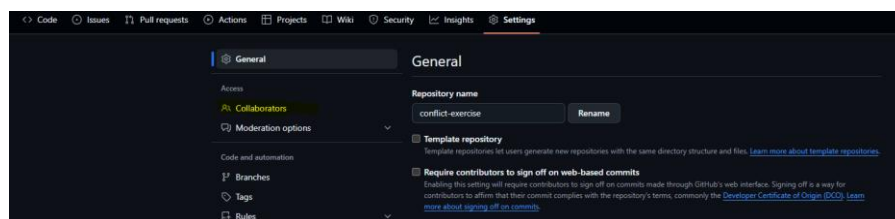


- Configurar el repositorio: Ponerle un nombre, elegir privado.

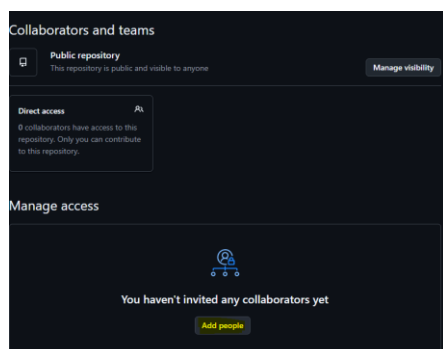


Esto asegura que el repositorio será privado y solo accesible para los colaboradores que tu elijas.

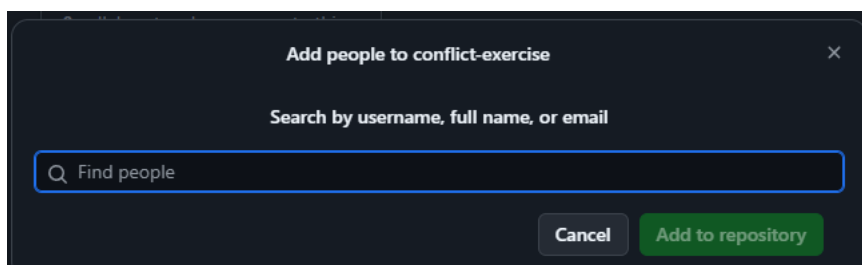
- ¿Cómo invitar a alguien a un repositorio privado en GitHub?
- Invitar a alguien a un repositorio privado en GitHub es un proceso sencillo, pero requiere permisos adecuados.
- Accede al repositorio, haz clic en la pestaña "Settings" del repositorio. Está en la parte superior del repositorio, junto a las pestañas como "Code" y "Issues".
- Seleccionar collaborators en el menú izquierdo



- Hacer click en "Add people"



- Ingresar el nombre del usuario, mail o nombre completo del colaborador



- Elegir el nivel de permiso: Read (solo lectura)

Write(Puede editar y hacer push)

Admin:Puede gestionar el repositorio(Invitar a otros, cambiar settings)

- Hacer click en “Add[usuario] to this Repository”
- ¿Qué es un repositorio público en GitHub?
- Un repositorio público en GitHub es un repositorio cuyo contenido es accesible a cualquier persona en Internet. A diferencia de un repositorio privado, que está restringido a un grupo específico de colaboradores, un repositorio público permite que cualquier persona pueda ver, clonar y, si tienen los permisos adecuados, contribuir al proyecto.

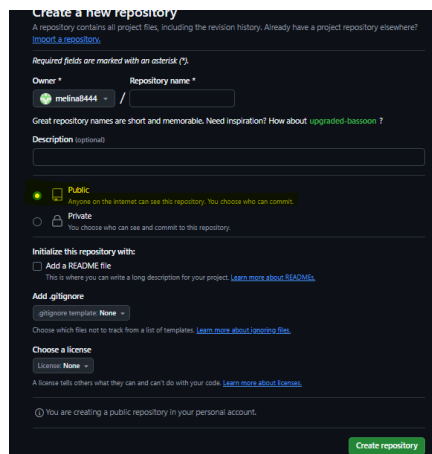
- ¿Cómo crear un repositorio público en GitHub?

Iniciar sesión: Se debe crear una cuenta en github.com y acceder a la cuenta.

Crear un repositorio: En la esquina superior derecha, en el signo +, hacer click, luego seleccionar “New Repository”.



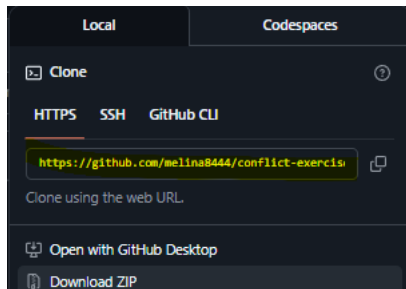
Configurar el repositorio: Ponerle un nombre, elegir publico.



Crear el repositorio: Haciendo click en el botón verde “Create Repository” .

- Esto asegura que el repositorio será público
- ¿Cómo compartir un repositorio público en GitHub?
- La forma más sencilla de compartir tu repositorio es proporcionar el enlace directo al mismo. Accede a tu repositorio, copia la URL de tu repositorio que se encuentra en un cuadro de texto que dice "Code":





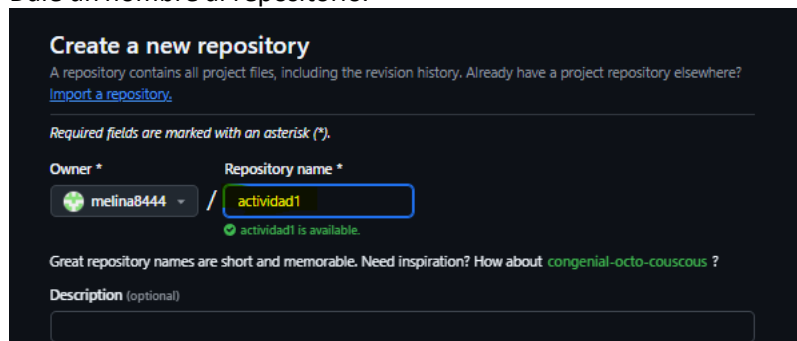
- Copiar la url
- Compartir el enlace.

2) Realizar la siguiente actividad:

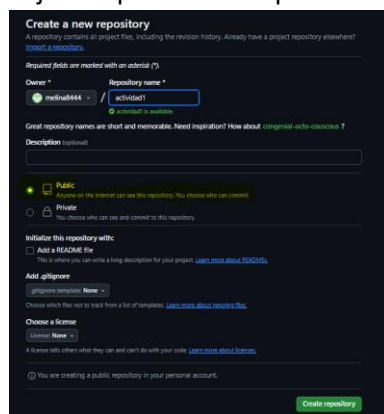
- Crear un repositorio haciendo click en “New”



- Dale un nombre al repositorio.



- Elije el repositorio sea público.



- Inicializa el repositorio con un archivo.
- En visual:

```
Usuario@DESKTOP-AU3N8BQ MINGW64 ~/Documents/utn_san_nicolas/CONTENIDO-MATERIAS/prog 1/actividad1
$ git init
Initialized empty Git repository in C:/Users/Usuario/Documents/utn_san_nicolas/CONTENIDO-MATERIAS/prog 1/actividad1/.git/
```

- Agregando un Archivo

- Crea un archivo simple, por ejemplo, "mi-archivo.txt".

```
Usuario@DESKTOP-AU3N8BQ MINGW64 ~/Documents/utn_san_nicolas/CONTENIDO-MATERIAS/prog 1/actividad1 (master)
```

- \$ echo "Este es mi archivo (Melina)" > mi-archivo.txt

- Realiza los comandos git add . y git commit -m "Agregando mi-archivo.txt" en la línea de comandos.

```
Usuario@DESKTOP-AU3N8BQ MINGW64 ~/Documents/utn_san_nicolas/CONTENIDO-MATERIAS/prog 1/actividad1 (master)
```

```
• $ git add .
```

```
Usuario@DESKTOP-AU3N8BQ MINGW64 ~/Documents/utn_san_nicolas/CONTENIDO-MATERIAS/prog 1/actividad1 (master)
```

```
• $ git status
```

```
On branch master
```

```
No commits yet
```

```
Changes to be committed:
```

```
(use "git rm --cached <file>..." to unstage)
```

```
new file:   mi-archivo.txt
```

```
Usuario@DESKTOP-AU3N8BQ MINGW64 ~/Documents/utn_san_nicolas/CONTENIDO-MATERIAS/prog 1/actividad1 (master)
```

```
• $ git commit -m "Agregando mi-archivo.txt"
```

```
[master (root-commit) 1840765] Agregando mi-archivo.txt
```

```
1 file changed, 1 insertion(+)
```

```
create mode 100644 mi-archivo.txt
```

```
Usuario@DESKTOP-AU3N8BQ MINGW64 ~/Documents/utn_san_nicolas/CONTENIDO-MATERIAS/prog 1/actividad1 (master)
```

```
• $ git branch -M main
```

```
Usuario@DESKTOP-AU3N8BQ MINGW64 ~/Documents/utn_san_nicolas/CONTENIDO-MATERIAS/prog 1/actividad1 (main)
```

```
• $ git remote add origin https://github.com/melina8444/actividad1.git
```

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

```
Usuario@DESKTOP-AU3N8BQ MINGW64 ~/Documents/utn_san_nicolas/CONTENIDO-MATERIAS/prog 1/actividad1 (main)
```

```
• $ git push -u origin main
```

```
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 (from 0)
```

```
To https://github.com/melina8444/actividad1.git
```

```
* [new branch]      main -> main
```

```
branch 'main' set up to track 'origin/main'.
```

- Creando Branchs

```
Usuario@DESKTOP-AU3N8BQ MINGW64 ~/Documents/utn_san_nicolas/CONTENIDO-MATERIAS/prog 1/actividad1 (main)
```

```
• $ git branch
```

```
* main
```

- Crear una Branch

```
Usuario@DESKTOP-AU3N8BQ MINGW64 ~/Documents/utn_san_nicolas/CONTENIDO-MATERIAS/prog 1/actividad1 (main)
```

```
• $ git branch rama1
```

```
Usuario@DESKTOP-AU3N8BQ MINGW64 ~/Documents/utn_san_nicolas/CONTENIDO-MATERIAS/prog 1/actividad1 (main)
```

```
• $ git branch
```

```
* main
```

```
rama1
```

- Realizar cambios o agregar un archivo

```
Usuario@DESKTOP-AU3N8BQ MINGW64 ~/Documents/utn_san_nicolas/CONTENIDO-MATERIAS/prog 1/actividad1 (main)
```

```
• $ echo "Este es un nuevo archivo (Agregando archivo en rama1)" > mi-nuevo-archivo.txt
```

- Subir la Branch

```
Usuario@DESKTOP-AU3N8BQ MINGW64 ~/Documents/utn_san_nicolas/CONTENIDO-MATERIAS/prog 1/actividad1 (main)
```

```
• $ git push origin rama1
```

```
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
```

```
remote:
```

```
remote: Create a pull request for 'rama1' on GitHub by visiting:
```

```
remote:      https://github.com/melina8444/actividad1/pull/new/rama1
```

```
remote:
```

```
To https://github.com/melina8444/actividad1.git
```

```
* [new branch]      rama1 -> rama1
```

Resolución de mi github:
<https://github.com/melina8444/actividad1.git>

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.

Resolución de mi github:

<https://github.com/melina8444/conflict-exercise.git>

