

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

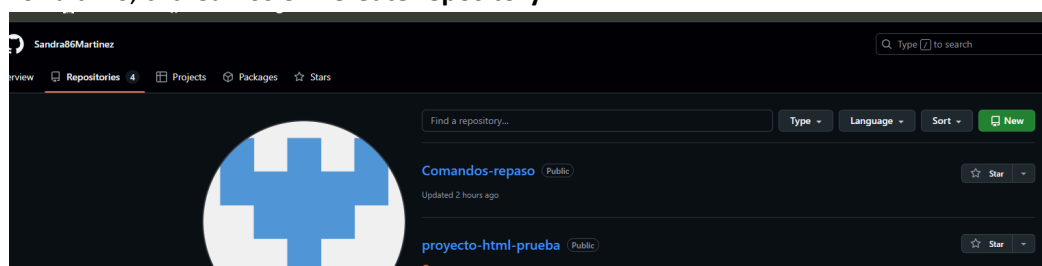
- 1) Contestar las siguientes preguntas utilizando las guías y documentación proporcionada (Desarrollar las respuestas) :

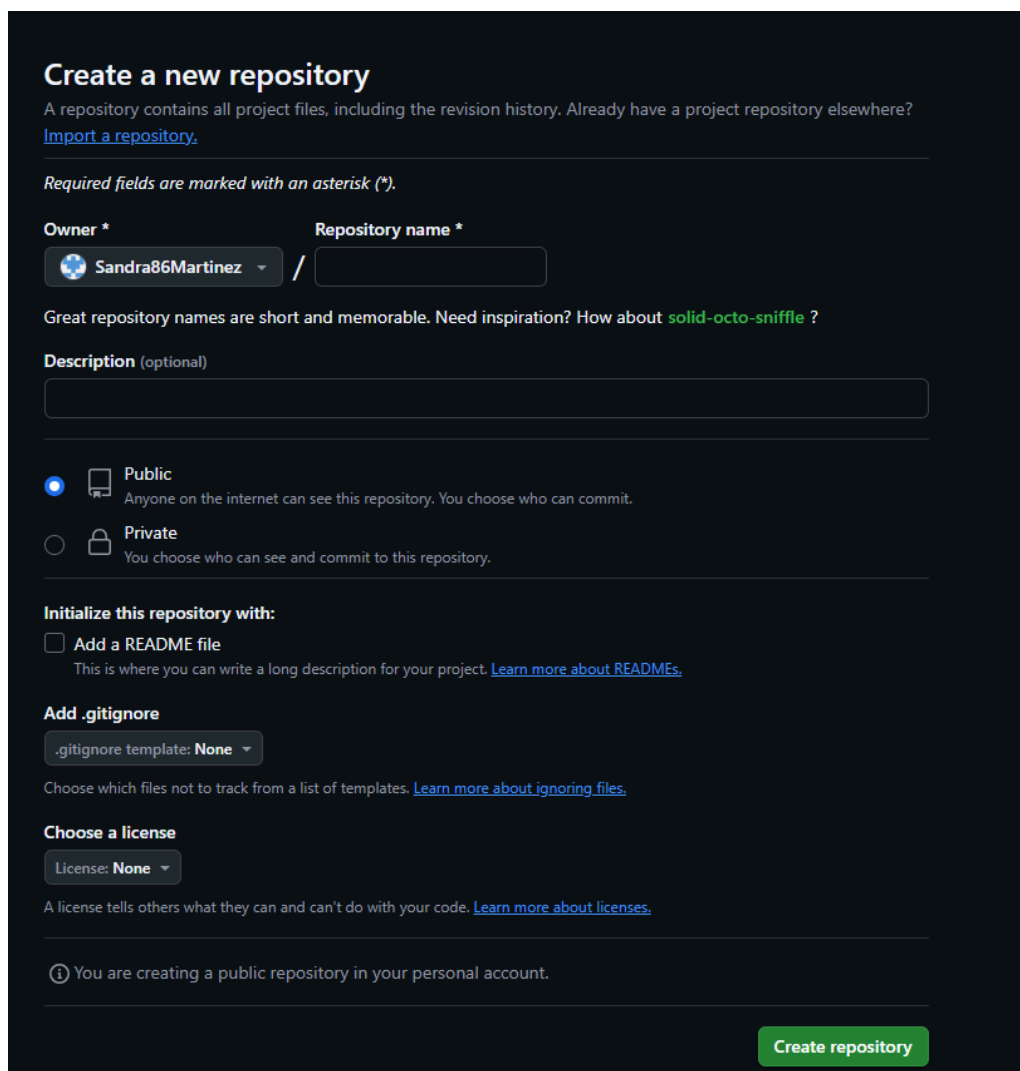
- **¿Qué es GitHub?**

GitHub es una plataforma en línea que permite almacenar proyectos que usan el sistema de control de versiones Git. Permite el trabajo colaborativo en distintos proyectos, como software de código abierto o cerrado. Los repositorios (lugar donde se guardan los contenidos de un proyecto) son almacenados en la nube. De esa manera, desarrolladores ubicados en diferentes partes del mundo pueden trabajar de forma simultánea, organizada y segura, compartiendo cambios, revisando código y mejorando proyectos en equipo.

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

- ❖ Dentro de la web (GitHub <https://github.com>) nos dirigimos al menú “Repositories”.
- ❖ Luego en la parte superior derecha clickeamos en “New”.
- ❖ Esto nos permite ingresar a una nueva pantalla y desde allí creamos el repo informando el nombre (**Repository name**) el cual es el campo obligatorio que debe ser completado si o si para poder crearlo. El resto es opcional como la descripción, si se necesita que sea público o privado, etc.
- ❖ Por último, clickeamos en “Create repository”





The screenshot shows the GitHub 'Create a new repository' page. At the top, it says 'Create a new repository' and provides a brief explanation of what a repository is. Below this, there's a note that required fields are marked with an asterisk (*). The form has two main sections: 'Owner' and 'Repository name', both with dropdown menus. The 'Owner' dropdown is set to 'Sandra86Martinez'. Below these fields, there's a suggestion for repository names: 'Great repository names are short and memorable. Need inspiration? How about **solid-octo-sniffle** ?'. The 'Description' field is optional and currently empty. There are two radio buttons for visibility: 'Public' (selected) and 'Private'. Below this, there's a section 'Initialize this repository with:' with a checkbox for 'Add a README file'. Underneath, it says 'Add .gitignore' with a dropdown menu set to 'None'. There's also a section 'Choose a license' with a dropdown menu set to 'None'. At the bottom, there's a green 'Create repository' button. A small information icon and text at the bottom left state: 'You are creating a public repository in your personal account.'

- **¿Cómo crear una rama en Git?**

Para crear una nueva rama en Git, abrimos la terminal y usamos el comando **"git branch <nombre>"**. Esto nos permite crear una nueva rama pero NO movernos a ella.

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

Para cambiarnos de una rama a otra, en la terminal, indicamos el comando: **git checkout <nombre>**.

- **¿Cómo fusionar ramas en Git?**

Para fusionar ramas (merge) por lo general, los cambios quedan en la rama principal. Lo primero que hacemos entonces es posicionarnos en la rama "main" o "master", con el comando **git checkout main ó master**. Luego, con el comando **git merge <rama>**, procedemos a la fusión. Esto permite que los cambios queden en la rama principal siempre y cuando no haya conflicto.

- **¿Cómo crear un commit en Git?**

Para crear un commit debemos utilizar en la terminal el comando: **git commit -m "mensaje"**. Guarda los archivos preparados como una nueva versión del proyecto.

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

Luego de realizar un commit, el envío se realiza con el comando: **git push origin nombre-de-la-rama**. "Git Push" sube los cambios. "Origin" es el nombre del repo remoto que por defecto se llama así. Por último, debemos identificar el nombre de la rama a subir.

- **¿Qué es un repositorio remoto?**

Un repositorio remoto es un espacio generalmente alojado en plataformas como GitHub y GitLab que permite sincronizar el trabajo con otros desarrolladores.

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

Para agregar un escritorio remoto a Git, debemos primero tener creado el repo en GitHub. Luego, con el comando **git remote add origin <https://github.com/usuario/XXXXXX>**, lo que hacemos es decirle a Git que “origin” será el repo remoto.

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

Para empujar los cambios, lo que hacemos es pushearlos. Utilizamos el comando **git push origin nombre-de-la-rama**. “Git push” es el comando que sube esos cambios a GitHub. “Origin” es el nombre del repo remoto que por defecto se llama así. Por último, debemos identificar el nombre de la rama a subir.

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

“Tirar” cambios es la traducción en español de “pull”. El comando descarga cambios recientes del repo remoto y los fusiona a la versión local. el comando es: **git pull origin nombre-de-la-rama**.

- **¿Qué es un fork de repositorio?**

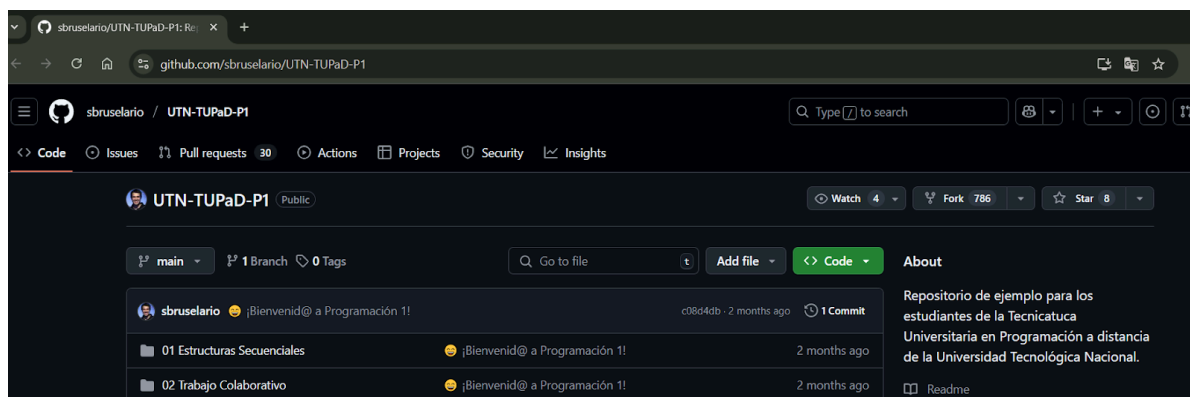
Un fork es una copia completa de un repo de GitHub que se crea dentro de nuestra propia cuenta. Permite:

- ❖ Probar cambios sin afectar el proyecto original.
- ❖ Proponer mejoras enviando los cambios mediante un pull request.
- ❖ Trabajar con repositorios a los que no tenés permisos directos.

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

Para crear un fork de un repo debemos:

1. Ingresá a un repositorio público de GitHub (por ejemplo, <https://github.com/otro-usuario/proyecto>).
2. Hacé clic en el botón Fork (esquina superior derecha).
3. Elegí tu cuenta para crear una copia del proyecto en tu propio GitHub.



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

Pull requests: Su función es proponer cambios a proyectos. Permiten revisiones, comentarios y aprobaciones antes de fusionar el código.

- ❖ Realizamos los cambios en una rama propia del repositorio
- ❖ Subimos esa rama al repositorio remoto: **git push origin nombre-de-la-rama**
- ❖ Ingresamos a GitHub y nos dirigimos al repo.
- ❖ GitHub nos muestra un botón que dice **"Compare & pull request"** (Comparar y hacer pull request). Clickeamos allí.
- ❖ Titulamos el cambio y realizamos una breve descripción.
- ❖ Verificamos que estamos solicitando fusionar nuestra rama con la rama principal.
- ❖ Hacemos click en "Create pull request"

- ¿Cómo aceptar una solicitud de extracción?

- ❖ Primero ingresamos a GitHub
- ❖ En la pestaña "Pull request" que se encuentra en la parte superior, hacemos click.
- ❖ Buscamos y seleccionamos el pull request que queremos revisar
- ❖ Cuando vemos todo OK y listo, realizamos el merge. Lo hacemos en el botón "Merge pull request". Esto fusiona la solicitud de extracción).
- ❖ Por último, confirmamos el merge.

- ¿Qué es un etiqueta en Git?

Una etiqueta en Git (o tag en inglés) es una marca que se coloca en un punto específico del historial de commits, generalmente para señalar una versión importante del proyecto, como una versión estable, un lanzamiento (v1.0, v2.1.3, etc.) o un hito importante.

- ¿Cómo crear una etiqueta en Git?

Con el comando: **git tag <nombre>**, creamos un tag simple sin mensaje.

- ¿Cómo enviar una etiqueta a GitHub?

Para enviar etiquetas a GitHub utilizamos el comando: **git push --tags**. Esta acción sube cambios incluyendo todos los tags.

- ¿Qué es un historial de Git?

El historial de Git es el registro completo de todos los cambios (commits) que se han hecho en un repo a lo largo del tiempo. Cada cambio incluye información sobre:

- ❖ El hash del commit (identificador único).
- ❖ El autor del cambio.
- ❖ La fecha del commit.
- ❖ El mensaje describe qué se hizo.
- ❖ Las modificaciones específicas en los archivos.

- ¿Cómo ver el historial de Git?

Para ver el historial, podemos usar los comandos:

git log: Ver commits (q para salir)

git log --oneline: Ver commits (una línea c/u)

git log --decorate --all --graph --oneline: Ver commits (graficado)

- ¿Cómo buscar en el historial de Git?

Para buscar en el historial de Git, podés usar varios comandos según lo que se requiera encontrar: palabras clave, archivos modificados, o incluso el autor del commit.

- ❖ Buscar por palabra clave en los mensajes de commit: **git log --grep="palabra-clave"**
- ❖ Ver historial de un archivo específico: **git log nombre-del-archivo**
- ❖ Ver cambios de un commit en detalle: **git show hash-del-commit**
- ❖ Historial resumido (una línea por commit): **git log --oneline**
- ❖ Buscar por autor: **git log --author="Nombre"**

- ¿Cómo borrar el historial de Git?

El borrado del historial de Git NO ES RECOMENDADO ya que sobrescribe todo el historial anterior y puede afectar a otros

desarrolladores si el repo es compartido. Los pasos a seguir son:

- ❖ git clone <https://github.com/usuario/repositorio.git>: de forma segura se clona el proyecto
- ❖ cd repositorio
- ❖ git checkout --orphan nueva-rama: creamos una rama temporal
- ❖ git add .: agregamos los archivos de nuevo
- ❖ git commit -m "comentario-inicio-nuevo"
- ❖ git branch -D <nombre-rama>: eliminamos la rama anterior
- ❖ git branch -m main/master: renombramos la rama como main o master
- ❖ git push -f origin main/master: forzamos el push al remoto para sobrescribir el historial

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

Un repositorio privado en Git (como en GitHub, GitLab o Bitbucket) es un proyecto que puede ser trabajado o visto por nosotros como administradores o usuario principal y por quienes nosotros autorizamos.

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

- ❖ Dentro de la web (GitHub <https://github.com>) nos dirigimos al menú “Repositories”.
- ❖ Luego en la parte superior derecha clickeamos en “New”.
- ❖ Esto nos permite ingresar a una nueva pantalla y desde allí creamos el repo informando el nombre (Repository name) el cual es el campo obligatorio que debe ser completado si o si para poder crearlo.
- ❖ Allí es donde elegimos a quienes pueden ver y commitear ese nuevo repo al elegir la opción: **PRIVATE**.
- ❖ Por último, clickeamos 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 * Sandra86Martinez / **Repository name ***

Great repository names are short and memorable. Need inspiration? How about **cuddly-robot** ?

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
.gitignore template: None

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

Choose a license
License: None

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

i You are creating a private repository in your personal account.

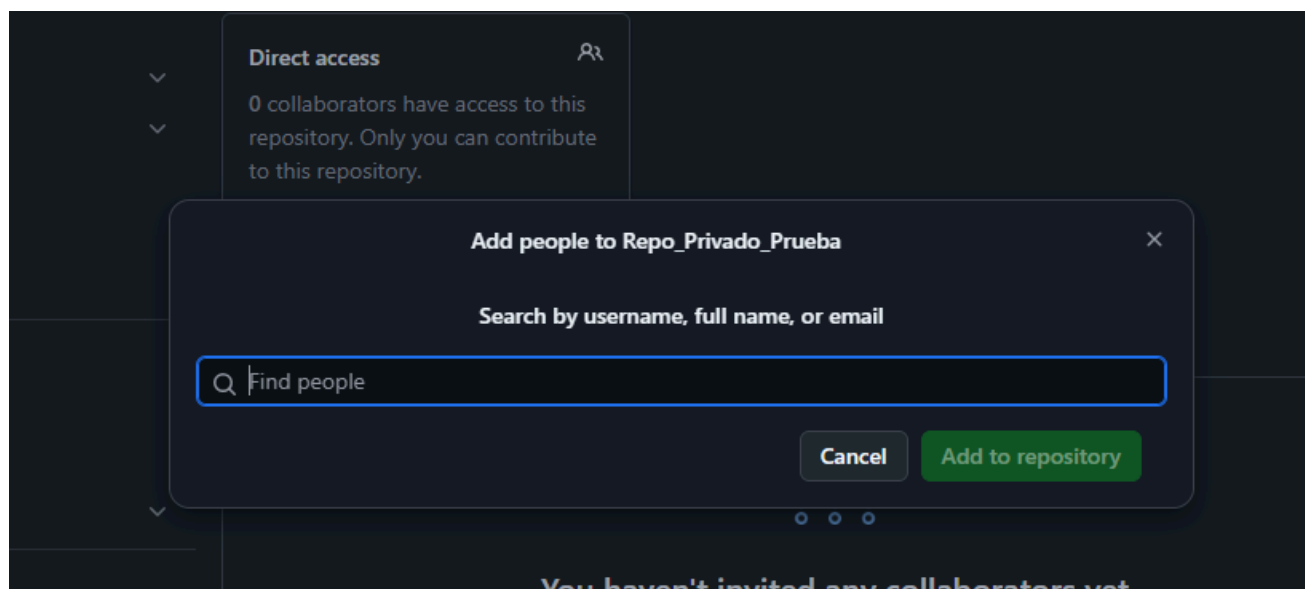
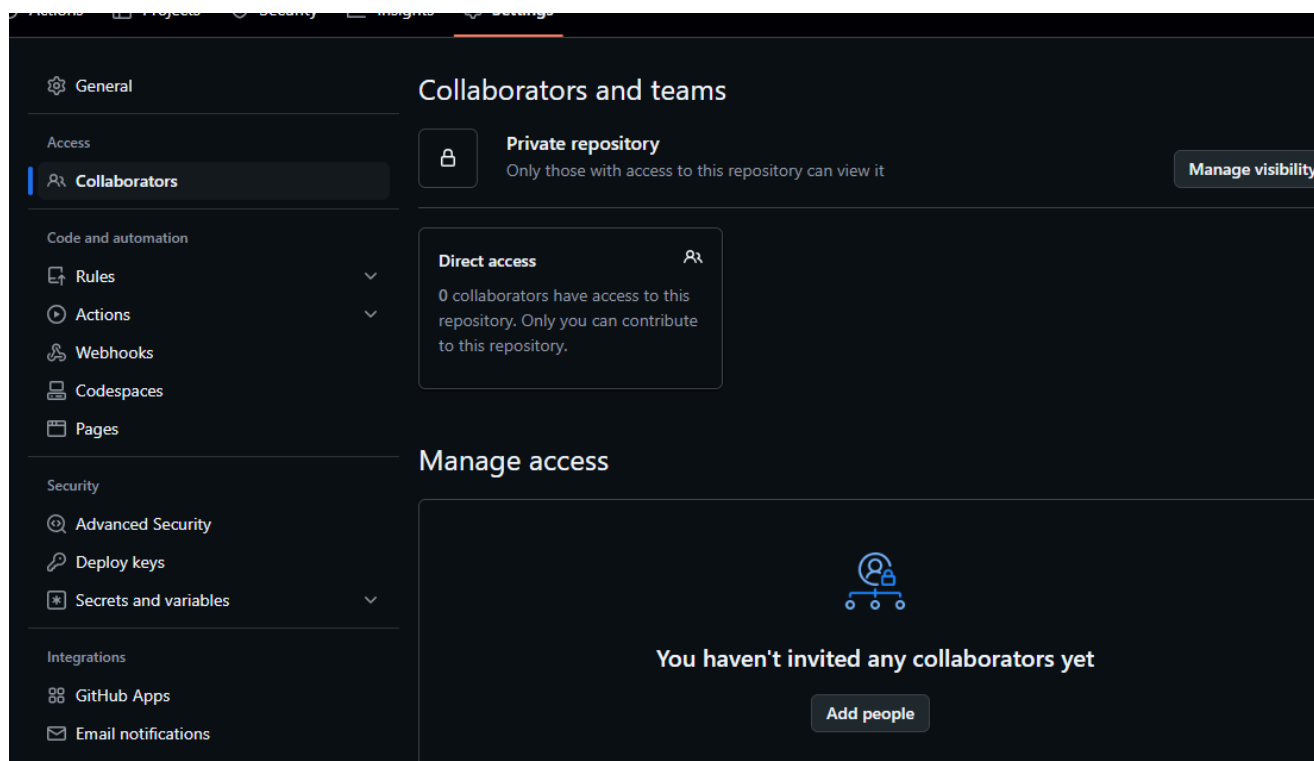
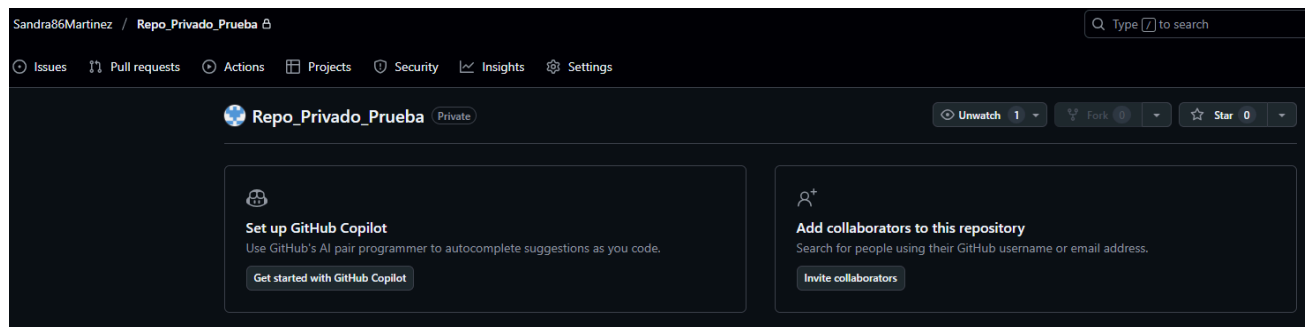
Create repository

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

Como usuarios y administradores de un repo privado podemos invitar a otros desarrolladores a él realizando los siguientes

pasos:

- ❖ Ingresamos al repo privado
- ❖ Luego nos dirigimos a la opción **“Invite collaborators”**
- ❖ En la opción **“Add people”**
- ❖ Por último clickeamos en: **“Add to repository”**

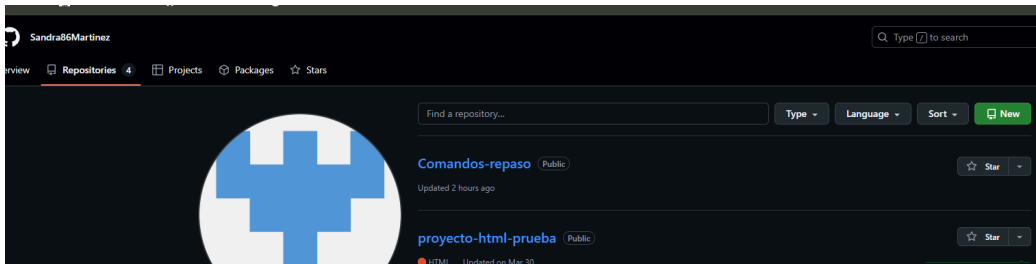


- ¿Qué es un repositorio público en GitHub?

Un repo público se diferencia del privado ya que permite a cualquier usuario ingresar al contenido. No necesita ser

agregado. Se utilizan comúnmente para ver repo con códigos abiertos.

- ¿Cómo crear un repositorio público en GitHub?
- ❖ Dentro de la web (GitHub <https://github.com>) nos dirigimos al menú “**Repositories**”.
- ❖ Luego en la parte superior derecha clickeamos en “**New**”.
- ❖ Esto nos permite ingresar a una nueva pantalla y desde allí creamos el repo informando el nombre (**Repository name**) el cual es el campo obligatorio que debe ser completado si o si para poder crearlo.
- ❖ Elegimos la opción “**Public**”
- ❖ Por último, clickeamos 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 * Sandra86Martinez / **Repository name ***

Great repository names are short and memorable. Need inspiration? How about **solid-octo-sniffle** ?

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
.gitignore template: **None**

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

Choose a license
License: **None**

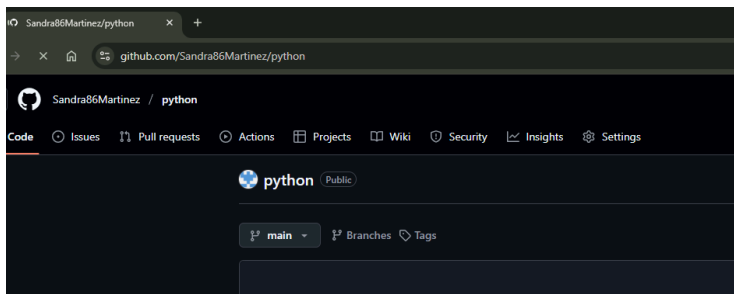
A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

☐ You are creating a public repository in your personal account.

Create repository

- ¿Cómo compartir un repositorio público en GitHub?

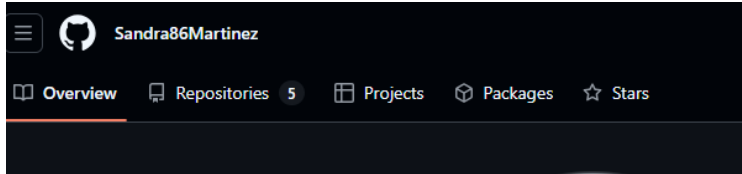
Para compartir un repo público, basta con copiar la URL de la barra de dirección del navegador, y enviarla por el medio que se desee.



2) Realizar la siguiente actividad:

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

❖ Dentro de la web Github, me dirijo a la solapa “Repositories”



❖ Luego, clicleo en el icono “New” ubicado en la parte superior derecha de la pantalla



- ❖ En la siguiente pantalla procedemos a informar el nombre del repo en el campo “Repository name” porque es obligatorio y elegimos la opción “Public”. Significa que cualquier usuario puede visualizar el proyecto.
- ❖ Por último, nos dirigimos a la parte inferior derecha “Create repository” para dar terminar la creación del repo.

- ❖ Por último, inicializamos el repo desde Git. Para esto, armé una carpeta y dentro de ella desarrollé un archivo con un código de prueba. El código lo trabajé y lo guardé desde Visual Studio Code.
- ❖ Luego, en VSC (Visual Studio Code) abrí una nueva terminal desde el menú superior: **Terminal- New Terminal**.
- ❖ Allí, ingresé los siguientes comandos

C:\Users\oscar\Desktop\RepoTP2>**git init (inicializamos el repo)**

Initialized empty Git repository in C:/Users/oscar/Desktop/RepoTP2/.git/

C:\Users\oscar\Desktop\RepoTP2>**git status (me muestra si hay cambios)**

On branch master

No commits yet

Untracked files:

(use "git add <file>..." to include in what will be committed)

repo_prueba_1.py

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

C:\Users\oscar\Desktop\RepoTP2>**git add .(agregamos los archivos al "staging area" para luego hacer un commit)**

C:\Users\oscar\Desktop\RepoTP2>**git commit -m "Prueba de archivo para inicializar un repo" (realizamos el primer commit del archivo con su respectivo comentario)**

[master (root-commit) 0af955a] Prueba de archivo para inicializar un repo

1 file changed, 6 insertions(+)

create mode 100644 repo_prueba_1.py

C:\Users\oscar\Desktop\RepoTP2>**git remote add origin <https://github.com/Sandra86Martinez/Repo-TP2-GIT-GITHUB.git> (vinculamos el proyecto local con el repo remoto)**

C:\Users\oscar\Desktop\RepoTP2>**git branch -M main (Renombramos la rama local)**

C:\Users\oscar\Desktop\RepoTP2>**git push -u origin main (pusheamos el archivo al repo remoto, o sea, lo subimos)**

Enumerating objects: 3, done.

Counting objects: 100% (3/3), done.

Delta compression using up to 4 threads

Compressing objects: 100% (2/2), done.

Writing objects: 100% (3/3), 364 bytes | 182.00 KiB/s, done.

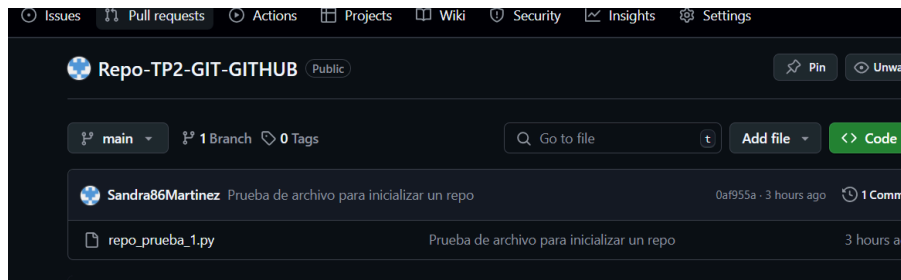
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)

To <https://github.com/Sandra86Martinez/Repo-TP2-GIT-GITHUB.git>

* [new branch] main -> main

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

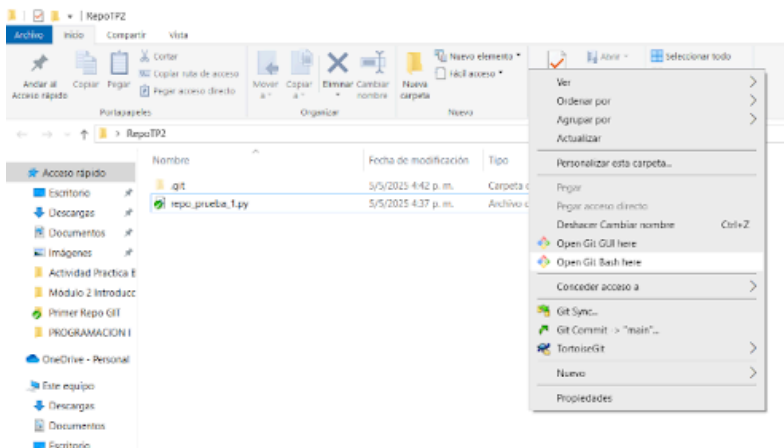
❖ Visualizamos en GitHub el repo archivo inicializado y subido



• 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).

❖ Desde la carpeta donde se encuentra el archivo creado (**mi-archivo.txt**), nos ubicamos en cualquier lado en blanco y con click derecho abrimos la opción: **Open Git Bush Here**.



❖ Una vez abierta la terminal, procedemos a subir el archivo. Yo usaré el mismo repo que había creado en el punto anterior.

❖ Seguí cada instrucción de la consigna y el archivo ya se visualiza en el repo remoto.

```
oscar@DESKTOP-0R60B0M MINGW64 ~/Desktop/RepoTP2 (main)
$ git add .

oscar@DESKTOP-0R60B0M MINGW64 ~/Desktop/RepoTP2 (main)
$ git commit -m "Agregando mi-archivo.txt"
[main bc54b29] Agregando mi-archivo.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 mi-archivo.txt

oscar@DESKTOP-0R60B0M MINGW64 ~/Desktop/RepoTP2 (main)
$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 313 bytes | 313.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Sandra86Martinez/Repo-TP2-GIT-GITHUB.git
0af955a..bc54b29 main -> main
```



2

Programación I



TECNICATURA UNIVERSITARIA EN PROGRAMACIÓN A DISTANCIA

• Creando Branchs

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

- ❖ Primero, procedí a modificar un código ya creado para mostrar este procedimiento de realizar cambios en una nueva rama
- ❖ Luego, con el comando **"git branch nueva-rama-tp"** procedí a crear la nueva rama
- ❖ Me cambié a esa nueva rama con el comando **"git checkout nueva-rama-tp"**
- ❖ Luego, con los comandos **"git add ."** y **"git commit -m 'Agregué una variable nueva llamada materia al archivo .py'"** procedí a preparar el escenario de cambios para después "comitear" la modificación.
- ❖ Por último, hice un push para subir esa rama a mi repo remoto utilizando los comandos: **"git push origin nueva-rama-tp"**

```
oscar@DESKTOP-OR60B0M MINGW64 ~/Desktop/RepoTP2 (main)
$ git branch nueva-rama-tp

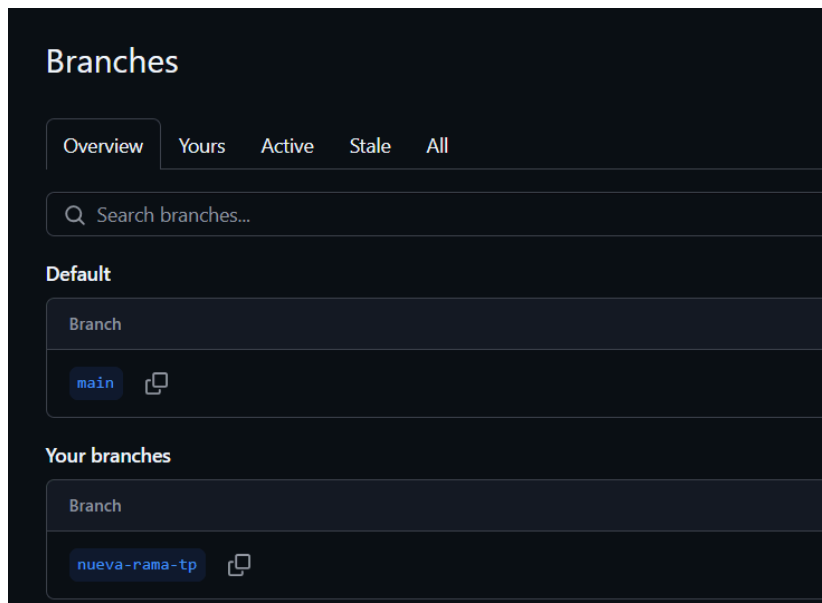
oscar@DESKTOP-OR60B0M MINGW64 ~/Desktop/RepoTP2 (main)
$ git checkout nueva-rama-tp
Switched to branch 'nueva-rama-tp'

oscar@DESKTOP-OR60B0M MINGW64 ~/Desktop/RepoTP2 (nueva-rama-tp)
$ git add .

oscar@DESKTOP-OR60B0M MINGW64 ~/Desktop/RepoTP2 (nueva-rama-tp)
$ git commit -m "Agregué una variable nueva llamada materia al archivo .py"
[nueva-rama-tp 7de93ea] Agregué una variable nueva llamada materia al archivo .py
1 file changed, 2 insertions(+), 1 deletion(-)

oscar@DESKTOP-OR60B0M MINGW64 ~/Desktop/RepoTP2 (nueva-rama-tp)
$ git push origin nueva-rama-tp
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 424 bytes | 424.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'nueva-rama-tp' on GitHub by visiting:
remote:   https://github.com/Sandra86Martinez/Repo-TP2-GIT-GITHUB/pull/new/nueva-rama-tp
remote:
To https://github.com/Sandra86Martinez/Repo-TP2-GIT-GITHUB.git
 * [new branch]      nueva-rama-tp -> nueva-rama-tp
```

❖ En GitHub quedó ya subida la nueva rama:



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".

The image shows the 'Create a new repository' form in GitHub. The form includes the following fields and options:

- Owner ***: A dropdown menu showing 'Sandra86Martinez'.
- Repository name ***: A text input field containing 'conflict-exercise'. Below it, a green checkmark indicates 'conflict-exercise is available'.
- Description (optional)**: A text input field.
- Visibility**: Two radio buttons. 'Public' is selected, with the description 'Anyone on the internet can see this repository. You choose who can commit.' 'Private' is unselected, with the description 'You choose who can see and commit to this repository.'
- Initialize this repository with:** A checkbox labeled 'Add a README file' is checked. Below it, a note says 'This is where you can write a long description for your project. [Learn more about READMEs.](#)'
- Add .gitignore**: A dropdown menu showing '.gitignore template: None'. Below it, a note says 'Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)'
- Choose a license**: A dropdown menu showing 'License: None'. Below it, a note says 'A license tells others what they can and can't do with your code. [Learn more about licenses.](#)'
- This will set**: A note saying 'This will set `main` as the default branch. Change the default name in your [settings.](#)'
- Footer**: A note saying 'You are creating a public repository in your personal account.'
- Create repository**: A green button at the bottom right.

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
```

```
oscar@DESKTOP-0R60B0M MINGW64 ~/Desktop
$ git clone https://github.com/Sandra86Martinez/conflict-exercise.git
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.

oscar@DESKTOP-0R60B0M MINGW64 ~/Desktop
$ 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"
```

```
oscar@DESKTOP-0R60B0M MINGW64 ~/Desktop/conflict-exercise (main)
$ git checkout -b feature-branch
Switched to a new branch 'feature-branch'

oscar@DESKTOP-0R60B0M MINGW64 ~/Desktop/conflict-exercise (feature-branch)
$ git status
On branch feature-branch
nothing to commit, working tree clean

oscar@DESKTOP-0R60B0M MINGW64 ~/Desktop/conflict-exercise (feature-branch)
$ git add README.md

oscar@DESKTOP-0R60B0M MINGW64 ~/Desktop/conflict-exercise (feature-branch)
$ git commit -m "Added a line in feature-branch"
[feature-branch 8bf7f53] Added a line in feature-branch
1 file changed, 2 insertions(+), 1 deletion(-)
```



- 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"
```

```
oscar@DESKTOP-0R60B0M MINGW64 ~/Desktop/conflict-exercise (feature-branch)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

oscar@DESKTOP-0R60B0M MINGW64 ~/Desktop/conflict-exercise (main)
$ git add README.md

oscar@DESKTOP-0R60B0M MINGW64 ~/Desktop/conflict-exercise (main)
$ git commit -m "Added a line in main branch"
[main 27a5bb1] Added a line in main branch
1 file changed, 2 insertions(+), 1 deletion(-)

oscar@DESKTOP-0R60B0M MINGW64 ~/Desktop/conflict-exercise (main)
```

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.

```
oscar@DESKTOP-0R60B0M MINGW64 ~/Desktop/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.

oscar@DESKTOP-0R60B0M MINGW64 ~/Desktop/conflict-exercise (main|MERGING)
```

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.

```
=====
```

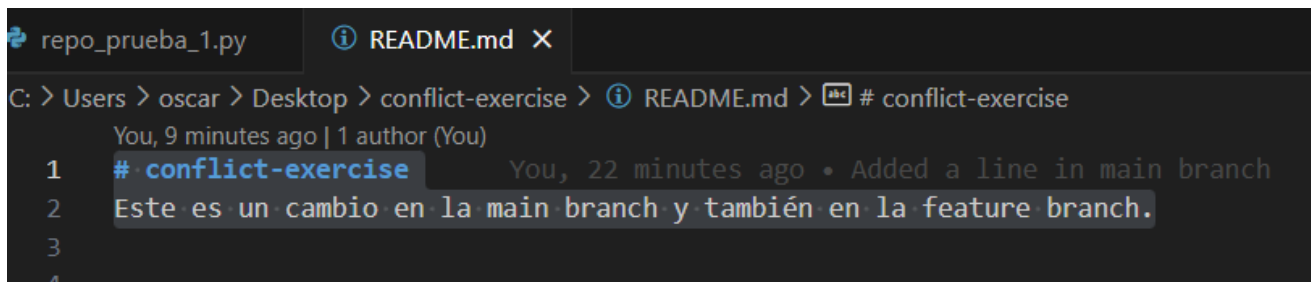
Este es un cambio en la feature branch.

```
>>>>>> 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"
```



```
oscar@DESKTOP-0R60B0M MINGW64 ~/Desktop/conflict-exercise (main|MERGING)
$ git add README.md

oscar@DESKTOP-0R60B0M MINGW64 ~/Desktop/conflict-exercise (main|MERGING)
$ git commit -m "Resolved merge conflict"
[main 39c9fa4] 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
```

```
oscar@DESKTOP-0R60B0M MINGW64 ~/Desktop/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), 827 bytes | 118.00 KiB/s, done.
Total 9 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/Sandra86Martinez/conflict-exercise.git
db99bb3..39c9fa4 main -> main

oscar@DESKTOP-0R60B0M MINGW64 ~/Desktop/conflict-exercise (main)
```



TECNICATURA UNIVERSITARIA EN PROGRAMACIÓN A DISTANCIA

- También sube la feature-branch si deseas:

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

```
oscar@DESKTOP-0R60B0M MINGW64 ~/Desktop/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/Sandra86Martinez/conflict-exercise/pull/new/feature-branch
remote:
To https://github.com/Sandra86Martinez/conflict-exercise.git
 * [new branch]      feature-branch -> 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.

Commits

main

Commits on May 5, 2025

Resolved merge conflict	Sandra86Martinez committed 6 minutes ago	39c9fa4		
Added a line in main branch	Sandra86Martinez committed 35 minutes ago	27a5bb1		
Added a line in feature-branch	Sandra86Martinez committed 42 minutes ago	8bf7f53		
Initial commit	Sandra86Martinez authored 1 hour ago	Verified db99bb3		