TRABAJO PRACTICO N° 2 GIT y GITHUB

1) ACTIVIDADES – RESOLUCION

1. ¿Qué es GitHub?

Es una plataforma en línea para alojamiento y control de desarrollo de software y colaboración en proyectos de versiones. Donde los programadores gestionan y comparten código, el cual realizan seguimiento de cambios en los archivos y colaborando en equipo utilizando Git.

2. ¿Cómo crear un repositorio en GitHub?

El primer paso es iniciar GitHub: Si aún no se tiene una cuenta, debemos registrarnos. Configuramos nuestro perfil y estamos listos para alojar nuestro repositorios.

En el botón "New": En tu perfil, busca un botón que diga "New" o "Nuevo" (suele estar en la parte superior derecha).

Completamos los detalles del repositorio:

a. **Nombre del repositorio**: Dale un nombre único y relevante.
b. **Descripción (opcional)**: Agrega una breve descripción sobre el proyecto.
c. **Visibilidad**: Elige entre "Público" (accesible para todos) o "Privado" (solo tú y las personas que invites pueden verlo).

Opciones adicionales:

a. Agregar un archivo README (útil para documentar tu proyecto).
b. Elegir una licencia (indica cómo otros pueden usar tu código).
c. Configurar un archivo `.gitignore`.

Crea el repositorio: Haz clic en "Create repository" o "Crear repositorio" y estamos listos para trabajar.

3. ¿Cómo crear una rama en Git?

Iniciamos sesión en Git Bash o línea de comandos y navegamos al directorio de tu proyecto usando `cd`.

Escribe el comando:

```
git branch nombre-de-tu-rama
```

Sustituimos `nombre-de-tu-rama` por el nombre que desees para tu nueva rama.

Cambia a esa rama para empezar a trabajar en ella con:

git checkout nombre-de-tu-rama

4. ¿Cómo cambiar a una rama en Git?

Consultamos las ramas disponibles en el repositorio con:

git branch

Cambiamos a la rama deseada: Utilizando el comando:

git checkout nombre-de-la-rama

Si la rama que queremos usar se llama `desarrollo`, se escribe:

git checkout desarrollo

Después de cambiar de rama, podemos verificar tu rama activa con:

git Branch

Denotando con un asterisco * la rama activa.

5. ¿Cómo fusionar ramas en Git?

Es el proceso de combinar los cambios de una rama a otra. Por ejemplo:

```
Git checkout rama-destino
git merge rama-origen
```

Esto combinará los cambios de la rama-origen en la rama-destino.

6. ¿Cómo crear un commit a GitHub?

Un commit en Git es esencial para guardar los cambios que has realizado en un proyecto. Aquí están los pasos principales:

```
git add archivo
```

si queremos agregar todos los cambios de todos los archivos en tu directorio:

git add .

Podemos crear un commit con un mensaje descriptivo, luego de agregar los cambios:

git commit -m "Agregamos cambios en las líneas 456 a 470"

7. ¿Cómo enviar el commit a GitHub?

Enviar un commit a GitHub implica "subir" tus cambios al repositorio remoto.

Nos aseguramos estar en la rama correcta, con el siguiente comando:

```
git Branch
git push origin nombre-de-la-rama
```

Si estás en la rama principal (main), podrías usar:

git push origin main

8. ¿Qué es un repositorio remoto?

Un repositorio remoto en Git es una versión del repositorio que se encuentra almacenada en un servidor en la nube o en otro equipo. Los repositorios remotos te permiten colaborar con otros desarrolladores, ya que todos pueden acceder al mismo repositorio, sincronizar cambios y trabajar en equipo. Implica la colaboración con otras personas, enviando y recibiendo datos cada vez que se comparte los repositorios.

La gestión de repositorios remotos incluye saber añadir, eliminar los repositorios, gestionar ramas remotas, definir si deben rastrearse o no y más.

9. ¿Cómo agregar un repositorio remoto a Git?

Usa el siguiente comando para añadir el repositorio remoto. Reemplaza:

https://github.com/usuario/repositorio.git

con la URL de tu repositorio remoto.

```
git remote add origin https://github.com/usuario/repositorio.git
```

Confirma que el repositorio remoto se ha añadido correctamente con:

git remote -v

Después de configurar el repositorio remoto, puedes subir tus cambios utilizando:

git push -u origin rama

si estás trabajando en la rama `main`:

git push -u origin main

10. ¿Cómo empujar cambios a un repositorio remoto?

Antes que nada se obtiene los últimos cambios del repositorio remoto para evitar conflictos:

```
git pull origin nombre_de_la_rama
```

empujando los cambios en el escritorio remoto:

```
git push origin nombre_de_la_rama
```

11. ¿Cómo tirar los cambios de un repositorio remoto?

Usamos los comandos antes vistos:

```
git pull origin nombre_de_la_rama
```

```
git push origin nombre_de_la_rama
```

12. ¿Qué es un fork de un repositorio?

Un **fork** de un repositorio es una copia del repositorio original que se crea en tu cuenta dentro de una plataforma, como GitHub. Es útil cuando quieres trabajar en un proyecto existente pero no tienes acceso directo para modificarlo, como ocurre con proyectos de código abierto.

13. ¿Cómo crear un fork de un repositorio?

Vamos al repositorio que deseamos copiar. Hacemos click en el botón **Fork** en la parte superior derecha.

El repositorio se copiará en tu cuenta para que puedas trabajar en él.

Una vez que tengas tu fork, puedes clonar esa copia en tu computadora con:

```
git clone https://github.com/tu-usuario/repositorio-forkeado.git
```

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

Enviar una solicitud de extracción (pull request) a un repositorio en GitHub es una forma de contribuir a un proyecto colaborativo. Aquí te dejo los pasos básicos para hacerlo:

Hacer un fork del repositorio: Ve al repositorio al que quieres contribuir y haz clic en el botón “Fork” en la esquina superior derecha. Esto crea una copia del repositorio en tu cuenta de GitHub.

Clona el fork en tu computadora: Usa el comando:

git clone https://github.com/tu_usuario/nombre_del_repositorio.git

Esto descarga el repositorio en tu máquina local.

Creamos una rama para los cambios, cambiamos a una nueva rama para trabajar en tus modificaciones. Por ejemplo:

git checkout -b mi-rama-de-cambios

Realizamos los cambios, hacemos los ajustes o agregados necesarios en el código. Una vez terminados, guardamos y realizamos un commit con:

git add .

git commit -m "Descripción de los cambios realizados"

Empuja tus cambios a tu fork en GitHub, subimos los cambios desde tu rama local al repositorio remoto en tu cuenta:

git push origin mi-rama-de-cambios

15. ¿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 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 master de su repositorio local remoto, puede clickear en merge pull requests y de esta manera sumará a su repositorio los cambios que hizo un usuario (en modo de ayuda).

16. ¿Qué es una etiqueta en Git?

En Git, una **etiqueta** es un marcador que se utiliza para señalar un punto específico en el historial del repositorio, generalmente para identificar una versión importante o estable del proyecto, como un lanzamiento oficial.

17. ¿Cómo crear una etiqueta en Git?

Crear una etiqueta en Git es un proceso sencillo y muy útil para marcar versiones importantes en tu proyecto. Aquí tienes los pasos para hacerlo:

Si necesitas solo un marcador simple, puedes usar:

```
git tag nombre_etiqueta
```

Para incluir más detalles como un mensaje y metadatos (autor, fecha, etc.), utiliza:

```
git tag -a nombre_etiqueta -m "Descripción de la etiqueta"
```

Si queremos ver el historial de etiquetas:

```
git tag
```

18. ¿Cómo enviar una etiqueta en GitHub?

Para subir una etiqueta específica al repositorio remoto en GitHub, usa:

```
git push origin nombre_etiqueta
```

Si deseas subir todas las etiquetas que hayas creado, utiliza:

```
git push origin –tags
```

19. ¿Qué es un historial de Git?

En Git, el **historial** es el registro completo de todos los cambios realizados en un repositorio, organizado por *commits*. Cada commit representa una versión específica del proyecto, incluyendo quién hizo el cambio, cuándo lo hizo y un mensaje que describe el propósito del cambio.

20. ¿Cómo ver el historial de Git?

```
git log
```

Este comando muestra detalles como:

- El hash único del commit.
- El autor del cambio.
- La fecha del commit.
- El mensaje asociado al commit.

21. ¿Cómo buscar en el historial de Git?

Es una habilidad esencial para localizar cambios específicos o commits relevantes en un proyecto.

```
git log --grep="palabra_clave"
```

git log --author="nombre_autor"

Visualización gráfica:

git log --graph --oneline –all

22. ¿Cómo borrar el historial de Git?

Borrar el historial de Git puede referirse a distintas acciones, como eliminar los commits previos del repositorio o limpiar el historial de cambios.

Este reinicio es irreversible:

rm -rf .git

git init

23. ¿Qué es un repositorio privado?

Un **repositorio privado** en GitHub es un espacio de almacenamiento para proyectos en el que los contenidos (como el código, archivos y el historial de cambios) están protegidos y solo accesibles para las personas que tú autorices. A diferencia de los repositorios públicos, los repositorios privados no son visibles para el público general.

24. ¿Cómo crear un repositorio privado en GitHub?

- **Inicia sesión en GitHub:** Ve a github.com e inicia sesión con tu cuenta.
- **Crea un nuevo repositorio:**
  - Haz clic en el botón verde **"New"** o ve a la pestaña de “Repositories” y selecciona "New".
  - Alternativamente, usa el enlace directo: Crear nuevo repositorio.
- **Configura el repositorio:**
  - Dale un nombre a tu repositorio en el campo “Repository name”.
  - Opcionalmente, escribe una descripción del proyecto.
- **Marca la opción "Private":**
  - En la sección “Visibility”, selecciona **"Private"** para asegurarte de que solo las personas que autorices tendrán acceso.
- **Configura opciones adicionales (opcional):**

- Puedes inicializar el repositorio con un archivo `README.md`, un archivo `.gitignore` o una licencia.

- **Crea el repositorio:**
  - Haz clic en el botón **"Create repository"**.

25. ¿Cómo invitar a alguien a un repositorio privado en GitHub?

1. **Accede al repositorio privado:** Inicia sesión en GitHub y ve al repositorio privado en el que deseas agregar a alguien.
2. **Abre la configuración del repositorio:**
   - Haz clic en la pestaña **"Settings"** (Configuración) en la parte superior del repositorio.
3. **Accede a la sección de colaboradores:**
   - Desplázate hacia abajo hasta la sección **"Collaborators and teams"** o similar y haz clic en **"Collaborators"**.
4. **Invita a un colaborador:**
   - En el campo de búsqueda, introduce el nombre de usuario o la dirección de correo electrónico de la persona que deseas invitar.
   - Haz clic en **"Add collaborator"** (Agregar colaborador).
5. **Envía la invitación:**
   - GitHub enviará una invitación a la persona. Esta debe aceptar la invitación antes de obtener acceso al repositorio.

Configura los permisos del colaborador:

Por defecto, los colaboradores tendrán acceso de escritura. Si necesitas ajustar los permisos (como lectura, escritura o administrador), puedes hacerlo desde la misma sección de configuración.

26. ¿Qué es un repositorio público en GitHub?

Un **repositorio público** en GitHub es un espacio donde puedes almacenar y compartir proyectos de código abierto u otros contenidos de manera accesible para todo el mundo. Cualquier usuario de GitHub, incluso aquellos que no estén registrados, puede ver, clonar y descargar un repositorio público.

27. ¿Cómo crear un repositorio público en GitHub?

1. **Inicia sesión en tu cuenta de GitHub:** Ve a GitHub y accede a tu cuenta.
2. **Crea un nuevo repositorio:**
   -

- Haz clic en el botón verde **"New"** (Nuevo) que aparece en la sección de tus repositorios o ve directamente a Crear nuevo repositorio.

3. **Completa la configuración inicial:**
   - **Repository name:** Escribe un nombre para tu repositorio.
   - **Description (opcional):** Proporciona una descripción breve de tu proyecto para que otros entiendan de qué trata.
   - **Visibility:** Selecciona la opción **"Public"** (Público) para hacer que el repositorio sea accesible a cualquier usuario.
4. **Inicializa el repositorio (opcional):**
   - Puedes elegir agregar un archivo `README.md`, un `.gitignore` o una licencia desde el inicio.
5. **Haz clic en "Create repository":**
   - Esto creará el repositorio público y estará listo para que subas tus archivos o realices tus cambios.

28. ¿Cómo compartir un repositorio público en Git Hub?

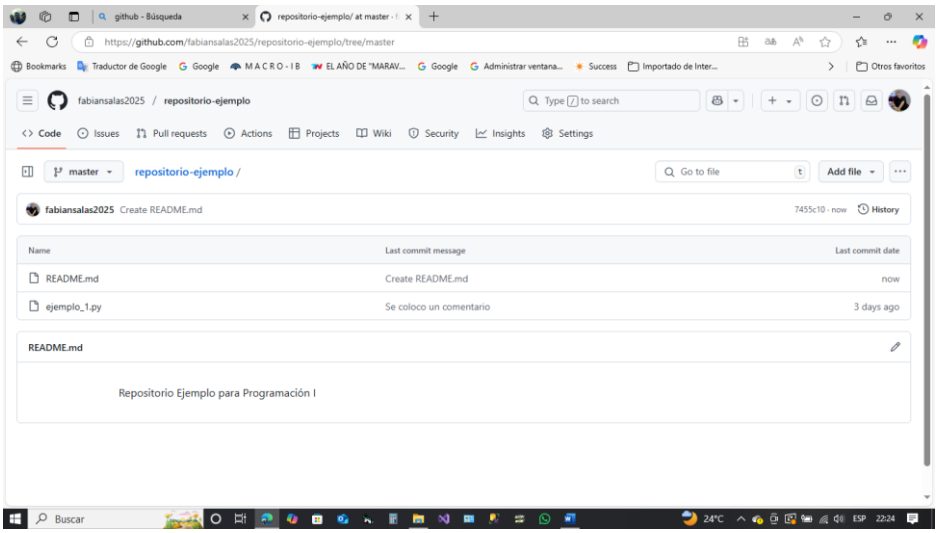
1. **Accede al repositorio público:** Inicia sesión en GitHub y abre el repositorio público que deseas compartir.
2. **Copia la URL del repositorio:**
   - En la barra de navegación de tu navegador, copia el enlace del repositorio. Por ejemplo, podría ser algo como:

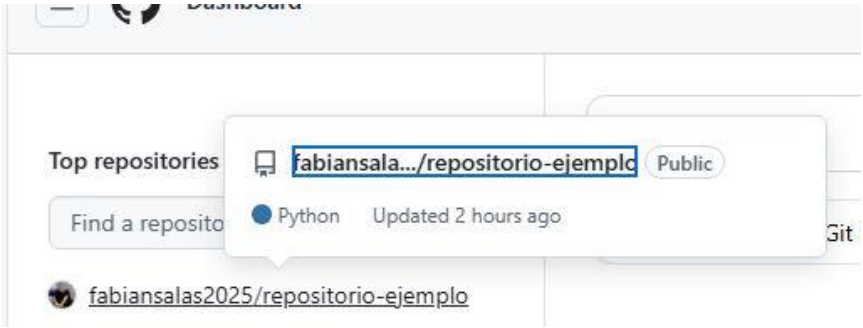
   https://github.com/tu_usuario/nombre_repositorio

3. **Envía el enlace:**
   - Comparte el enlace por correo electrónico, mensajería instantánea, redes sociales o donde prefieras. Cualquiera que reciba el enlace podrá acceder al repositorio sin necesidad de autorización.

2) ACTIVIDADES – RESOLUCION

- Creamos un repositorio

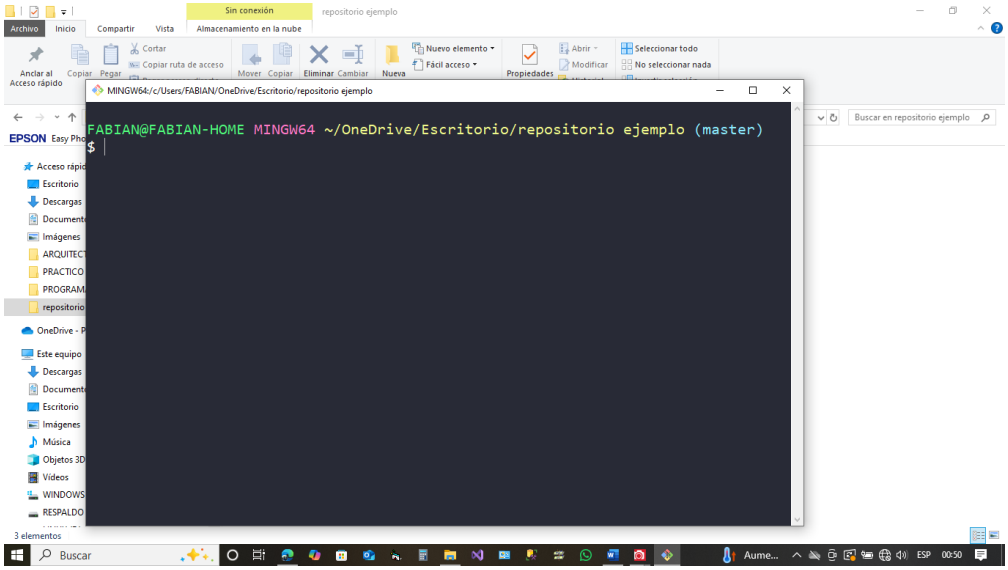




- Se creó el archivo

C:\Users\FABIAN\OneDrive\Escritorio\repositorio ejemplo\ejemplo_1.py

Se abre git Basch.



Se inicializa el directorio

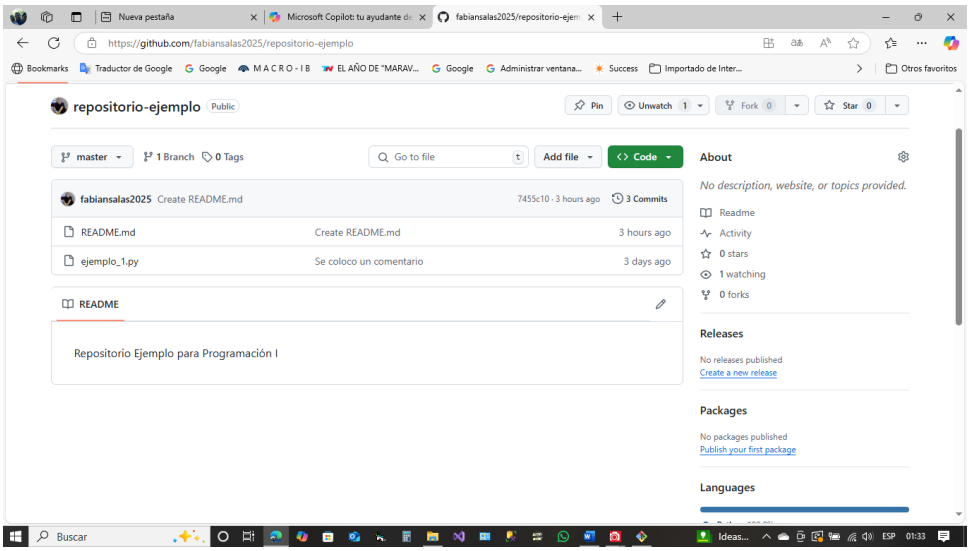
Git init

Git add .

Git commit -m “Se agrega archivo ejemplo_1.py”

Git remote add origin https:/github.com/fabiansalas2025/repositorio-ejemplo.git

Git push origin master



Creando Branch

Git Branch nueva_rama

Git checkout nueva_rama

Git Branch

master

*nueva_rama

Dir

Ejemplo_1.py

Modificamos archivo.

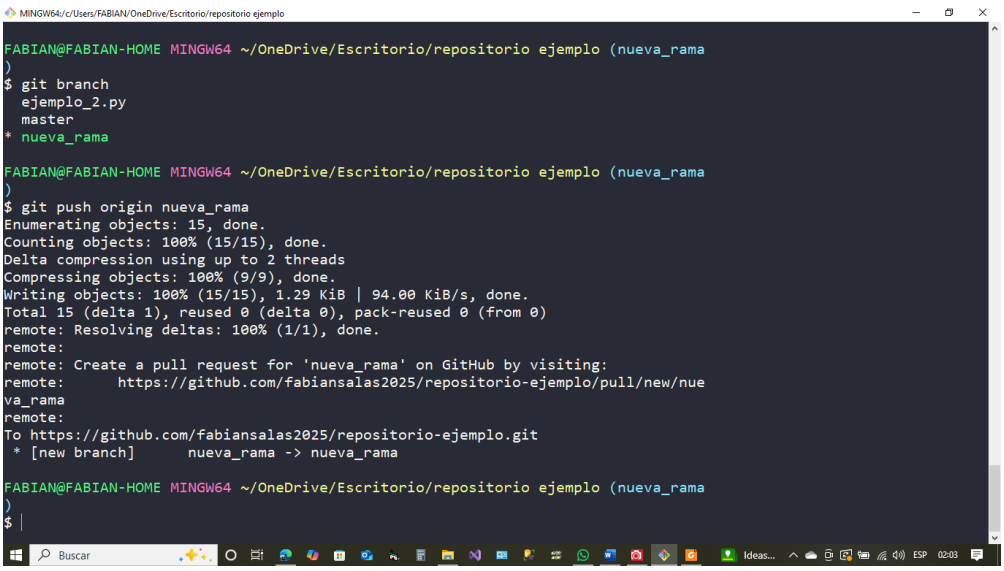
Git add .

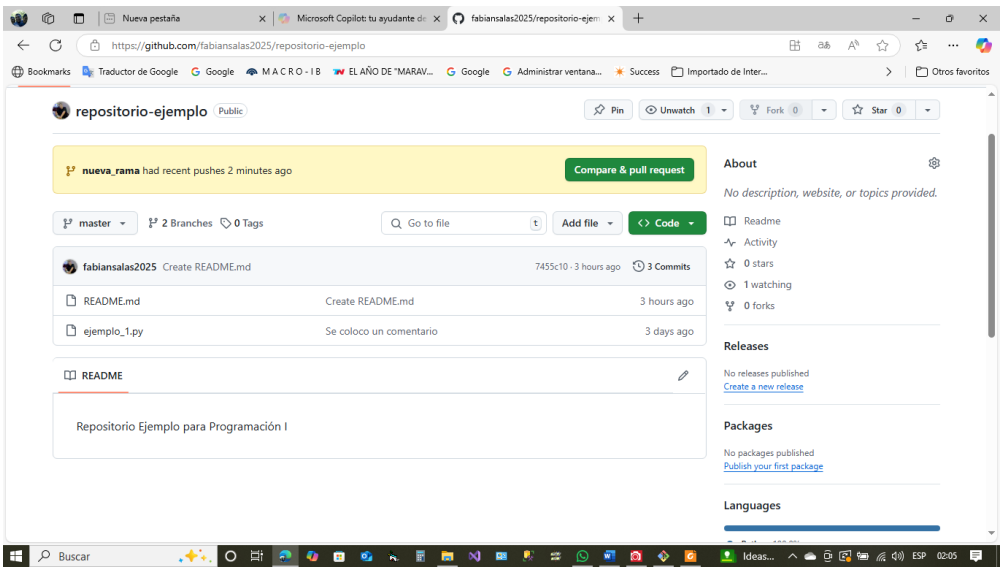
Git commit -m “Se modificó archivo”

Git status

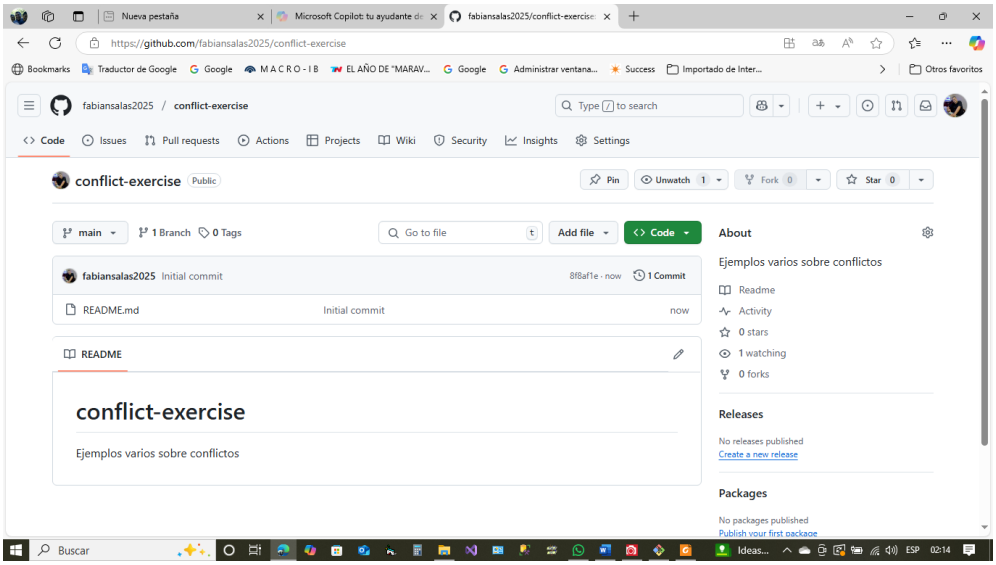
Se modificó archivo en nueva_rama

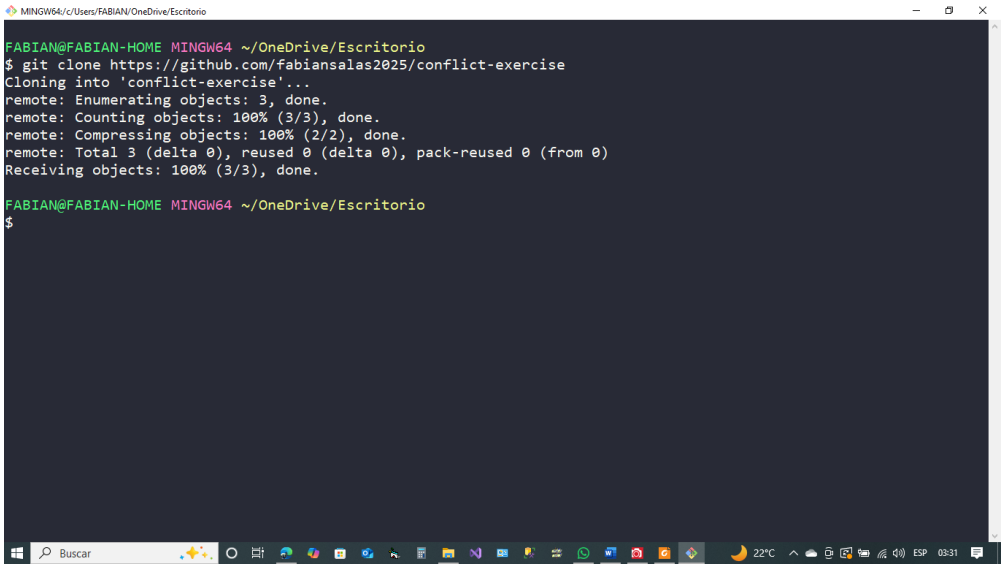
Se sube nueva rama


MINGW64:/c/Users/FABIAN/OneDrive/Escritorio/repositorio ejemplo
FABIAN@FABIAN-HOME MINGW64 ~/OneDrive/Escritorio/repositorio ejemplo (nueva_rama
)
$ git branch
ejemplo_2.py
master
* nueva_rama
FABIAN@FABIAN-HOME MINGW64 ~/OneDrive/Escritorio/repositorio ejemplo (nueva_rama
)
$ git push origin nueva_rama
Enumerating objects: 15, done.
Counting objects: 100% (15/15), done.
Delta compression using up to 2 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (15/15), 1.29 KiB | 94.00 KiB/s, done.
Total 15 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), done.
remote:
remote: Create a pull request for 'nueva_rama' on GitHub by visiting:
remote: https://github.com/fabiansalas2025/repositorio-ejemplo/pull/new/nue
va_rama
remote:
To https://github.com/fabiansalas2025/repositorio-ejemplo.git
* [new branch] nueva_rama -> nueva_rama
FABIAN@FABIAN-HOME MINGW64 ~/OneDrive/Escritorio/repositorio ejemplo (nueva_rama
)
$


https://github.com/fabiansalas2025/repositorio-ejemplo
repositorio-ejemplo
Public
Pin
Unwatch 1
Fork 0
Star 0
nueva_rama had recent pushes 2 minutes ago
Compare & pull request
master
2 Branches
0 Tags
Go to file
Add file
Code
fabiansalas2025 Create README.md
7455c10 · 3 hours ago
3 Commits
README.md
Create README.md
3 hours ago
ejemplo_1.py
Se coloco un comentario
3 days ago
README
Repositorio Ejemplo para Programación I
About
No description, website, or topics provided.
Readme
Activity
0 stars
1 watching
0 forks
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package
Languages

3) Realizar la siguiente actividad





Cd conflicto-exercise

MINGW64:/c/Users/FABIAN/OneDrive/Escritorio/conflict-exercise

```
FABIAN@FABIAN-HOME MINGW64 ~/OneDrive/Escritorio/conflict-exercise (main)
$ git branch feature-branch

FABIAN@FABIAN-HOME MINGW64 ~/OneDrive/Escritorio/conflict-exercise (main)
$ git branch
  feature-branch
* main

FABIAN@FABIAN-HOME MINGW64 ~/OneDrive/Escritorio/conflict-exercise (main)
$ git checkout -b feature-branch
fatal: a branch named 'feature-branch' already exists

FABIAN@FABIAN-HOME MINGW64 ~/OneDrive/Escritorio/conflict-exercise (main)
$ git checkout feature-branch
Switched to branch 'feature-branch'

FABIAN@FABIAN-HOME MINGW64 ~/OneDrive/Escritorio/conflict-exercise (feature-branch)
$ git branch
* feature-branch
  main

FABIAN@FABIAN-HOME MINGW64 ~/OneDrive/Escritorio/conflict-exercise (feature-branch)
$ dir
README.md

FABIAN@FABIAN-HOME MINGW64 ~/OneDrive/Escritorio/conflict-exercise (feature-branch)
$
```

```
FABIAN@FABIAN-HOME MINGW64 ~/OneDrive/Escritorio/conflict-exercise (feature-branch)
$ dir
README.md

FABIAN@FABIAN-HOME MINGW64 ~/OneDrive/Escritorio/conflict-exercise (feature-branch)
$ git add README.md

FABIAN@FABIAN-HOME MINGW64 ~/OneDrive/Escritorio/conflict-exercise (feature-branch)
$ git commit -m "Se agregó un línea"
[feature-branch 2a0d917] Se agregó un línea
 1 file changed, 1 insertion(+)

FABIAN@FABIAN-HOME MINGW64 ~/OneDrive/Escritorio/conflict-exercise (feature-branch)
$
```

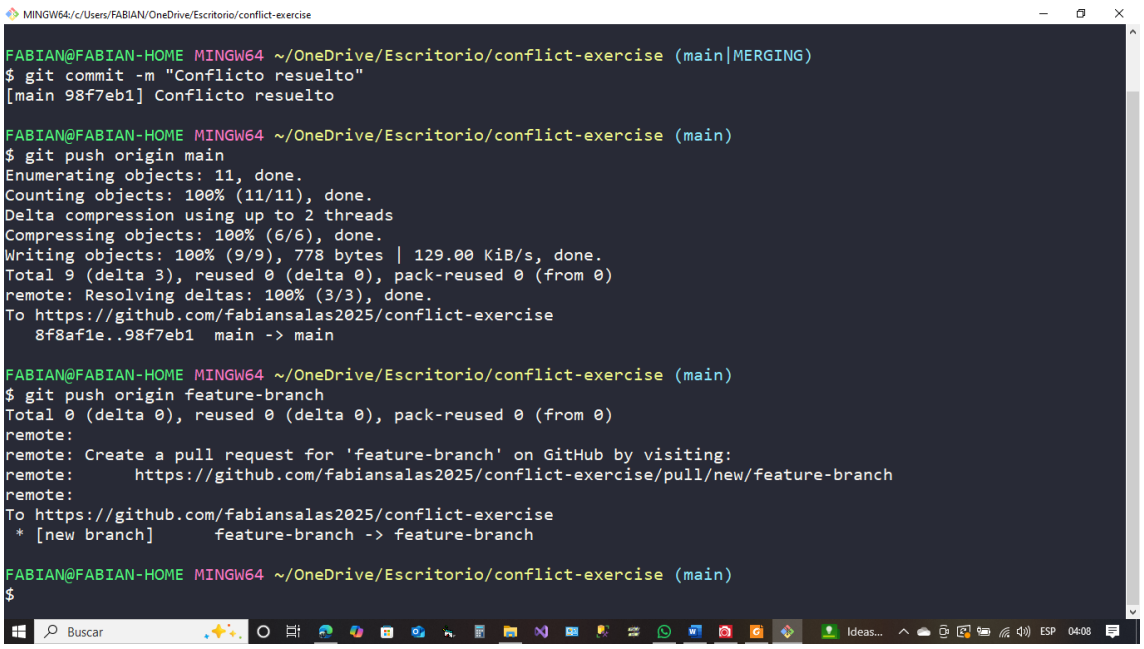
```
FABIAN@FABIAN-HOME MINGW64 ~/OneDrive/Escritorio/conflict-exercise (feature-branch)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

FABIAN@FABIAN-HOME MINGW64 ~/OneDrive/Escritorio/conflict-exercise (main)
$ git branch
  feature-branch
* main

FABIAN@FABIAN-HOME MINGW64 ~/OneDrive/Escritorio/conflict-exercise (main)
$
```

```
FABIAN@FABIAN-HOME MINGW64 ~/OneDrive/Escritorio/conflict-exercise (main)
$ git branch
  feature-branch
* main

FABIAN@FABIAN-HOME MINGW64 ~/OneDrive/Escritorio/conflict-exercise (main)
$ git add README.md

FABIAN@FABIAN-HOME MINGW64 ~/OneDrive/Escritorio/conflict-exercise (main)
$ git commit -m "Nueva linea en la rama main"
[main a93a13c] Nueva linea en la rama main
 1 file changed, 1 insertion(+)

FABIAN@FABIAN-HOME MINGW64 ~/OneDrive/Escritorio/conflict-exercise (main)
$
```

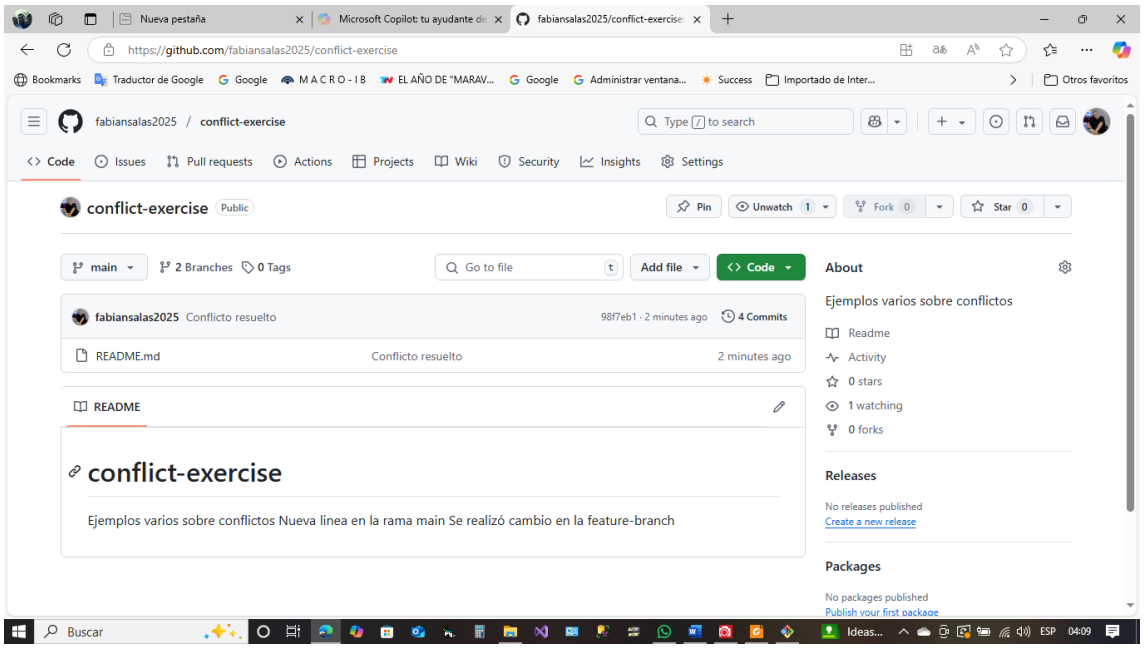
Hacemos merge y generamos conflicto:

```
FABIAN@FABIAN-HOME MINGW64 ~/OneDrive/Escritorio/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.

FABIAN@FABIAN-HOME MINGW64 ~/OneDrive/Escritorio/conflict-exercise (main|MERGING)
$
```

Resolvemos el conflicto:



Verificamos en GitHub:


Nueva pestaña
Microsoft Copilot: tu ayudante de
fabiansalas2025/conflict-exercise
https://github.com/fabiansalas2025/conflict-exercise
Bookmarks
Traductor de Google
Google
M A C R O - I B
EL AÑO DE "MARAV...
Google
Administrar ventana...
Success
Importado de Inter...
Otros favoritos
fabiansalas2025 / conflict-exercise
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
conflict-exercise
Public
Pin
Unwatch 1
Fork 0
Star 0
main
2 Branches
0 Tags
Go to file
Add file
Code
fabiansalas2025 Conflicto resuelto
98f7eb1 · 2 minutes ago
4 Commits
README.md
Conflicto resuelto
2 minutes ago
README
conflict-exercise
Ejemplos varios sobre conflictos Nueva linea en la rama main Se realizó cambio en la feature-branch
About
Ejemplos varios sobre conflictos
Readme
Activity
0 stars
1 watching
0 forks
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package
Buscar
Ideas...
ESP
04:09