Ejercicios

1.Clona un repositorio Git con múltiples ramas.

```
Albert@DESKTOP-F43VOVL MINGW64 /d/Varios/DS/actividad-5 (main)
$ git merge origin/feature
Updating 27a01fa..d64e078
Fast-forward
 main.py | 2 ++
 1 file changed, 2 insertions(+)

Albert@DESKTOP-F43VOVL MINGW64 /d/Varios/DS/actividad-5 (main)
$ git log --graph --oneline
* d64e078 (HEAD -> main, origin/feature) Update main.py
* 27a01fa (origin/main, origin/HEAD) Create main.py
* 2c7f292 Update README.md
* 34a6a71 Initial commit
```

Pregunta: ¿En qué situaciones recomendarías evitar el uso de git merge --ff? Reflexiona sobre las desventajas de este método.

Cuando se quiere mantener un historial claro de cuando han ocurrido fusiones ya que el merge fast forward puede dificultar la comprensión del historial de cambios.

2.Simula un flujo de trabajo de equipo.

```
Albert@DESKTOP-F43VOVL MINGW64 /d/Varios/DS/actividad-5 (main)
$ git merge --no-ff feature
Auto-merging main.py
CONFLICT (content): Merge conflict in main.py
Automatic merge failed; fix conflicts and then commit the result.

Albert@DESKTOP-F43VOVL MINGW64 /d/Varios/DS/actividad-5 (main|MERGING)
$ git add .

Albert@DESKTOP-F43VOVL MINGW64 /d/Varios/DS/actividad-5 (main|MERGING)
$ git commit -m "merge no-ff"
[main 768a470] merge no-ff

Albert@DESKTOP-F43VOVL MINGW64 /d/Varios/DS/actividad-5 (main)
$ git log --graph --oneline
*   768a470 (HEAD -> main) merge no-ff
|\
| * 1d84777 (feature) funcion saludo y despedida en feature
* | 2516098 funcion saludo
|/
* d64e078 (origin/feature) Update main.py
* 27a01fa (origin/main, origin/HEAD) Create main.py
* 2c7f292 Update README.md
* 34a6a71 Initial commit
```

Pregunta: ¿Cuáles son las principales ventajas de utilizar git merge --no-ff en un proyecto en equipo? ¿Qué problemas podrían surgir al depender excesivamente de commits de fusión?

Usar git merge –no-ff permite ver claramente cuando se ha hecho una fusión en la rama principal y permite revertir los cambios fácilmente si se encuentra algún error. Por otro lado si las ramas no se organizan correctamente y abusamos de los commits de fusión se hace más difícil usar otros comandos como git blame y

git log, además pueden haber más conflictos entre ramas y hacer más difícil la corrección de errores.

3.Crea múltiples commits en una rama

```
$ git merge --squash feature
Auto-merging main.py
CONFLICT (content): Merge conflict in main.py
Squash commit -- not updating HEAD
Automatic merge failed; fix conflicts and then commit the result.
```

```
Albert@DESKTOP-F43VOVL MINGW64 /d/Varios/DS/actividad-5 (main)
$ git add .

Albert@DESKTOP-F43VOVL MINGW64 /d/Varios/DS/actividad-5 (main)
$ git commit -m "merge squash"
[main dc9d80c] merge squash
 1 file changed, 29 insertions(+)

Albert@DESKTOP-F43VOVL MINGW64 /d/Varios/DS/actividad-5 (main)
$ git log --graph --oneline
* dc9d80c (HEAD -> main) merge squash
*   768a470 merge no-ff
|\
| * 1d84777 funcion saludo y despedida en feature
* | 2516098 funcion saludo
|/
* d64e078 (origin/feature) Update main.py
* 27a01fa (origin/main, origin/HEAD) Create main.py
* 2c7f292 Update README.md
* 34a6a71 Initial commit
```

Pregunta: ¿Cuándo es recomendable utilizar una fusión squash? ¿Qué ventajas ofrece para proyectos grandes en comparación con fusiones estándar?

Cuando en nuestra rama feature hacemos varios commits intermedios que no aportan realmente al historial, es recomendable usar la fusión squash. En proyectos grandes, esto facilita la navegación por el historial de cambios y permiten consolidar todo el trabajo de un feature en un único commit.

Resolver conflictos en una fusión non-fast-forward

1.Inicializa un nuevo repositorio:

```
Albert@DESKTOP-F43VOVL MINGW64 /d/Varios/DS
$ mkdir prueba-merge-conflict

Albert@DESKTOP-F43VOVL MINGW64 /d/Varios/DS
$ cd prueba-merge-conflict/

Albert@DESKTOP-F43VOVL MINGW64 /d/Varios/DS/prueba-merge-conflict
$ git init
Initialized empty Git repository in D:/Varios/DS/prueba-merge-conflict/.git/
```

2.Crea un archivo index.html y realiza un commit en la rama main:

```
Albert@DESKTOP-F43VOVL MINGW64 /d/Varios/DS/prueba-merge-conflict (main)
$ echo "<html><body><h1>Proyecto inicial CC3S2</h1></body></html>" > index.html
```

```
Albert@DESKTOP-F43V0VL MINGW64 /d/Varios/DS/prueba-merge-conflict (main)
$ git commit -m "commit inicial del index.html en main"
[main (root-commit) 7d8719f] commit inicial del index.html en main
 1 file changed, 1 insertion(+)
 create mode 100644 index.html
```

3.Crea y cambia a una nueva rama feature-update:

```
Albert@DESKTOP-F43V0VL MINGW64 /d/Varios/DS/prueba-merge-conflict (main)
$ git checkout -b feature-update
Switched to a new branch 'feature-update'
```

4.Edita el archivo y realiza un commit en la rama feature-update:

```
Albert@DESKTOP-F43V0VL MINGW64 /d/Varios/DS/prueba-merge-conflict (feature-updat
e)
$ echo "<p>.....</p>" >> index.html

Albert@DESKTOP-F43V0VL MINGW64 /d/Varios/DS/prueba-merge-conflict (feature-update)
$ git add index.html
warning: in the working copy of 'index.html', LF will be replaced by CRLF the next ti
ouches it

Albert@DESKTOP-F43V0VL MINGW64 /d/Varios/DS/prueba-merge-conflict (feature-update)
$ git commit -m "actualiza index.html"
[feature-update 8b2f0a4] actualiza index.html
 1 file changed, 1 insertion(+)
```

5.Regresa a la rama main y realiza una edición en el mismo archivo:

6.Fusiona la rama feature-update con --no-ff y observa el conflicto:

```
Albert@DESKTOP-F43V0VL MINGW64 /d/Varios/DS/prueba-merge-conflict (main)
$ git merge --no-ff feature-update
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
Automatic merge failed; fix conflicts and then commit the result.
```

7.Git detectará un conflicto en index.html. Abre el archivo y resuelve el conflicto. Elimina las líneas de conflicto generadas por Git (<<<<<<<, =======, >>>>>>>) y crea la versión final del archivo con ambos cambios:

8.Agrega el archivo corregido y completa la fusión:

```
Albert@DESKTOP-F43V0VL MINGW64 /d/Varios/DS/prueba-merge-conflict (main|MERGING)
$ git add index.html

Albert@DESKTOP-F43V0VL MINGW64 /d/Varios/DS/prueba-merge-conflict (main|MERGING)
$ git commit -m "merge"
[main 66521bc] merge
```

9.Verifica el historial para confirmar la fusión y el commit de resolución de conflicto:

```
Albert@DESKTOP-F43V0VL MINGW64 /d/Varios/DS/prueba-merge-conflict (main)
$ git log --graph --oneline
*   66521bc (HEAD -> main) merge
|\
| * 8b2f0a4 (feature-update) actualiza index.html
* | 0f2784a Nuevo footer en index.html
|/
* 7d8719f commit inicial del index.html en main
```

Preguntas:

- ¿Qué pasos adicionales tuviste que tomar para resolver el conflicto?

  Tuve que borrar los comentarios generados por git y editar el archivo en el que se hace el merge, además de hacer commit de los últimos cambios

- ¿Qué estrategias podrías emplear para evitar conflictos en futuros desarrollos colaborativos?

  Para evitar conflictos lo que podríamos hacer es definir un flujo de trabajo claro, de modo que todos los desarrolladores trabajen en ramas bien definidas y sepan cómo y cuando hacer merges, squash, etc, …

Ejercicio: Comparar los historiales con git log después de diferentes fusiones

1.Crea un nuevo repositorio y realiza varios commits en dos ramas:

```
Albert@DESKTOP-F43VOVL MINGW64 /d/Varios/DS/prueba-compare-merge (main)
$ git checkout -b feature-1
Switched to a new branch 'feature-1'

Albert@DESKTOP-F43VOVL MINGW64 /d/Varios/DS/prueba-compare-merge (feature-1)
$ ehco "Caracteristica 1 agregada" >> version.txt
bash: ehco: command not found

Albert@DESKTOP-F43VOVL MINGW64 /d/Varios/DS/prueba-compare-merge (feature-1)
$ echo "Caracteristica 1 agregada" >> version.txt

Albert@DESKTOP-F43VOVL MINGW64 /d/Varios/DS/prueba-compare-merge (feature-1)
$ git add version.txt
warning: in the working copy of 'version.txt', LF will be replaced by CRLF the ne
touches it

Albert@DESKTOP-F43VOVL MINGW64 /d/Varios/DS/prueba-compare-merge (feature-1)
$ git commit -m "Agregar caracteristica 1"
[feature-1 2e2b85a] Agregar caracteristica 1
 1 file changed, 1 insertion(+)

Albert@DESKTOP-F43VOVL MINGW64 /d/Varios/DS/prueba-compare-merge (feature-1)
$ git checkout main
Switched to branch 'main'

Albert@DESKTOP-F43VOVL MINGW64 /d/Varios/DS/prueba-compare-merge (main)
$ git checkout -b feature-2
Switched to a new branch 'feature-2'

Albert@DESKTOP-F43VOVL MINGW64 /d/Varios/DS/prueba-compare-merge (feature-2)
$ echo "Caracteristica 2 agregada" >> version.txt

Albert@DESKTOP-F43VOVL MINGW64 /d/Varios/DS/prueba-compare-merge (feature-2)
$ git add version.txt
warning: in the working copy of 'version.txt', LF will be replaced by CRLF the ne
touches it

Albert@DESKTOP-F43VOVL MINGW64 /d/Varios/DS/prueba-compare-merge (feature-2)
$ git commit -m "Se agrega caracteristica 2"
[feature-2 91df368] Se agrega caracteristica 2
 1 file changed, 1 insertion(+)
```

2.Fusiona feature-1 usando fast-forward:

```
Albert@DESKTOP-F43V0VL MINGW64 /d/Varios/DS/prueba-compare-merge (main)
$ git merge feature-1 --ff
Updating f598bc1..2e2b85a
Fast-forward
 version.txt | 1 +
 1 file changed, 1 insertion(+)
```

3. Fusiona feature-2 usando non-fast-forward:

```
Albert@DESKTOP-F43V0VL MINGW64 /d/Varios/DS/prueba-compare-merge (main)
$ git merge feature-2 --no-ff
Auto-merging version.txt
CONFLICT (content): Merge conflict in version.txt
Automatic merge failed; fix conflicts and then commit the result.

Albert@DESKTOP-F43V0VL MINGW64 /d/Varios/DS/prueba-compare-merge (main|MERGING)
$ git add .

Albert@DESKTOP-F43V0VL MINGW64 /d/Varios/DS/prueba-compare-merge (main|MERGING)
$ git commit -m "conflicto resuelto"
[main 46314c3] conflicto resuelto
```

4.Realiza una nueva rama feature-3 con múltiples commits y fusiónala con squash:

```
Albert@DESKTOP-F43V0VL MINGW64 /d/Varios/DS/prueba-compare-merge (feature-3)
$ echo "Caracteristica 3 paso 1" >> version.txt

Albert@DESKTOP-F43V0VL MINGW64 /d/Varios/DS/prueba-compare-merge (feature-3)
$ git add version.txt
warning: in the working copy of 'version.txt', LF will be replaced by CRLF the
touches it

Albert@DESKTOP-F43V0VL MINGW64 /d/Varios/DS/prueba-compare-merge (feature-3)
$ git commit -m "Caracteristica 3 paso 1"
[feature-3 49b649b] Caracteristica 3 paso 1
 1 file changed, 1 insertion(+)

Albert@DESKTOP-F43V0VL MINGW64 /d/Varios/DS/prueba-compare-merge (feature-3)
$ echo "Caracteristica 3 paso 2" >> version.txt

Albert@DESKTOP-F43V0VL MINGW64 /d/Varios/DS/prueba-compare-merge (feature-3)
$ git add version.txt
warning: in the working copy of 'version.txt', LF will be replaced by CRLF the
touches it

Albert@DESKTOP-F43V0VL MINGW64 /d/Varios/DS/prueba-compare-merge (feature-3)
$ git commit -m "Caracteristica 3 paso 2"
[feature-3 67dcfa2] Caracteristica 3 paso 2
 1 file changed, 1 insertion(+)

Albert@DESKTOP-F43V0VL MINGW64 /d/Varios/DS/prueba-compare-merge (feature-3)
$ git checkout main
Switched to branch 'main'

Albert@DESKTOP-F43V0VL MINGW64 /d/Varios/DS/prueba-compare-merge (main)
$ git merge --squash feature-3
Updating 46314c3..67dcfa2
Fast-forward
Squash commit -- not updating HEAD
 version.txt | 2 ++
 1 file changed, 2 insertions(+)

Albert@DESKTOP-F43V0VL MINGW64 /d/Varios/DS/prueba-compare-merge (main)
$ git commit -m "Agregar caracteristica 3 en un commit"
[main fab9176] Agregar caracteristica 3 en un commit
 1 file changed, 2 insertions(+)
```

5.Compara el historial de Git:

Historial Fast-forward

```
Albert@DESKTOP-F43VOVL MINGW64 /d/Varios/DS/prueba-compare-merge (main)
$ git log --graph --oneline --merges --first-parent --branches
* 46314c3 conflicto resuelto
```

Historial Non-fast-forward

```
Albert@DESKTOP-F43VOVL MINGW64 /d/Varios/DS/prueba-compare-merge (main)
$ git log --graph --oneline --merges
* 46314c3 conflicto resuelto
```

Historial con Squash

```
Albert@DESKTOP-F43VOVL MINGW64 /d/Varios/DS/prueba-compare-merge (main)
$ git log --graph --oneline --merges --decorate --all
* 46314c3 conflicto resuelto
```

Ejercicio: Usando fusiones automáticas y revertir fusiones

1.Inicializa un nuevo repositorio y realiza dos commits en main:

```
Albert@DESKTOP-F43VOVL MINGW64 /d/Varios/DS/prueba-auto-merge
$ git init
Initialized empty Git repository in D:/Varios/DS/prueba-auto-merge/.git/

Albert@DESKTOP-F43VOVL MINGW64 /d/Varios/DS/prueba-auto-merge (main)
$ echo "Linea 1" > file.txt

Albert@DESKTOP-F43VOVL MINGW64 /d/Varios/DS/prueba-auto-merge (main)
$ git add file.txt
warning: in the working copy of 'file.txt', LF will be replaced by CRLF the
 time Git touches it

Albert@DESKTOP-F43VOVL MINGW64 /d/Varios/DS/prueba-auto-merge (main)
$ git commit -m "Agrega linea 1"
[main (root-commit) 6d23fac] Agrega linea 1
 1 file changed, 1 insertion(+)
 create mode 100644 file.txt

Albert@DESKTOP-F43VOVL MINGW64 /d/Varios/DS/prueba-auto-merge (main)
$ echo "Linea 2" >> file.txt

Albert@DESKTOP-F43VOVL MINGW64 /d/Varios/DS/prueba-auto-merge (main)
$ git add file.txt
warning: in the working copy of 'file.txt', LF will be replaced by CRLF the
 time Git touches it

Albert@DESKTOP-F43VOVL MINGW64 /d/Varios/DS/prueba-auto-merge (main)
$ git commit -m "Agrega linea 2"
[main 34a5624] Agrega linea 2
 1 file changed, 1 insertion(+)
```

2.Crea una nueva rama auto-merge y realiza otro commit en file.txt:

```
Albert@DESKTOP-F43VOVL MINGW64 /d/Varios/DS/prueba-auto-merge (main)
$ git checkout -b auto-merge
Switched to a new branch 'auto-merge'

Albert@DESKTOP-F43VOVL MINGW64 /d/Varios/DS/prueba-auto-merge (auto-merge)
$ echo "Linea 3" >> file.txt

Albert@DESKTOP-F43VOVL MINGW64 /d/Varios/DS/prueba-auto-merge (auto-merge)
$ git add file.txt
warning: in the working copy of 'file.txt', LF will be replaced by CRLF the
 time Git touches it

Albert@DESKTOP-F43VOVL MINGW64 /d/Varios/DS/prueba-auto-merge (auto-merge)
$ git commit -m "Agrega linea 3"
[auto-merge 5329851] Agrega linea 3
 1 file changed, 1 insertion(+)
```

3. Vuelve a main y realiza cambios no conflictivos en otra parte del archivo:

```
Albert@DESKTOP-F43VOVL MINGW64 /d/Varios/DS/prueba-auto-merge (auto-merge)
$ git checkout main
Switched to branch 'main'

Albert@DESKTOP-F43VOVL MINGW64 /d/Varios/DS/prueba-auto-merge (main)
$ echo "Footer: Fin del archivo" >> file.txt

Albert@DESKTOP-F43VOVL MINGW64 /d/Varios/DS/prueba-auto-merge (main)
$ git add file.txt
warning: in the working copy of 'file.txt', LF will be replaced by CRLF th
 time Git touches it

Albert@DESKTOP-F43VOVL MINGW64 /d/Varios/DS/prueba-auto-merge (main)
$ git commit -m "Add footer al archivo file.txt"
[main 98a1176] Add footer al archivo file.txt
 1 file changed, 1 insertion(+)
```

4. Fusiona la rama auto-merge con main:

```
Albert@DESKTOP-F43VOVL MINGW64 /d/Varios/DS/prueba-auto-merge (main)
$ git merge auto-merge
Auto-merging file.txt
CONFLICT (content): Merge conflict in file.txt
Automatic merge failed; fix conflicts and then commit the result.

Albert@DESKTOP-F43VOVL MINGW64 /d/Varios/DS/prueba-auto-merge (main|MERGING)
$ git add .

Albert@DESKTOP-F43VOVL MINGW64 /d/Varios/DS/prueba-auto-merge (main|MERGING)
$ git commit -m "merge"
[main 9c0b4f5] merge
```

5. Git debería fusionar los cambios automáticamente sin conflictos.

6. Revertir la fusión: Si decides que la fusión fue un error, puedes revertirla:

```
Albert@DESKTOP-F43VOVL MINGW64 /d/Varios/DS/prueba-auto-merge (main)
$ git revert -m 1 HEAD
[main 4286dd5] Revert "error :("
 1 file changed, 5 insertions(+)
```

7. Verifica el historial

```
Albert@DESKTOP-F43V0VL MINGW64 /d/Varios/DS/prueba-auto-merge (main)
$ git log --graph --oneline
* 4286dd5 (HEAD -> main) Revert "error :("
* 90bc742 error :(
*   9c0b4f5 merge
|\
| * 5329851 (auto-merge) Agrega linea 3
* | 1088634 linea 2
* | 8bf6f91 otra correccion del footer
* | 0582956 correccion en el footer
* | 98a1176 Add footer al archivo file.txt
|/
* 34a5624 Agrega linea 2
* 6d23fac Agrega linea 1
```

¿Cuándo usarías un comando como git revert para deshacer una fusión?

Cuando haya muchos errores en el software debido a un merge con alguna característica y se necesite volver a una versión anterior inmediatamente

¿Qué tan útil es la función de fusión automática en Git?

Puede resultar útil cuando no hay conflictos

Ejercicio: Fusión remota en un repositorio colaborativo

1. Clona un repositorio remoto desde GitHub o crea uno nuevo:

```
Albert@DESKTOP-F43V0VL MINGW64 /d/Varios/DS
$ git clone git@github.com:Alkerg/fusion-remota.git
Cloning into 'fusion-remota'...
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.
```

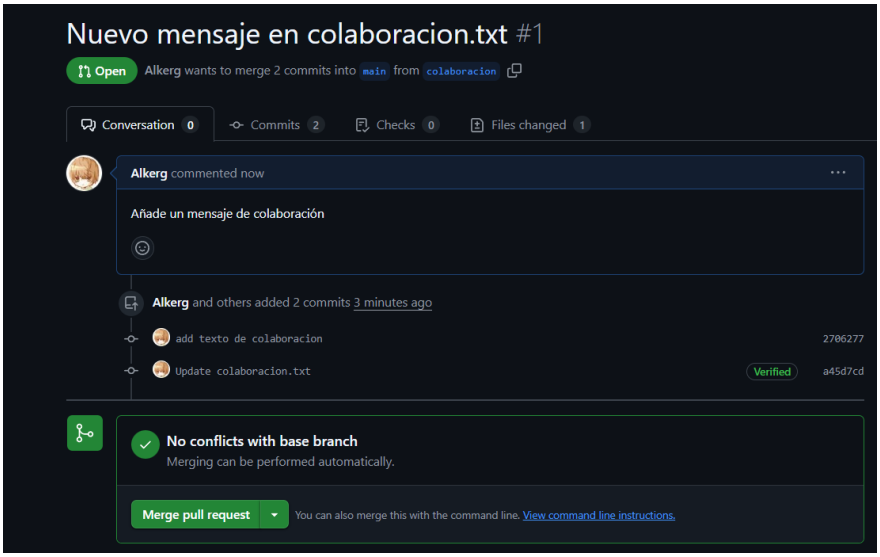
2. Crea una nueva rama colaboracion y haz algunos cambios:

```
Albert@DESKTOP-F43V0VL MINGW64 /d/Varios/DS/fusion-remota (main)
$ git checkout -b colaboracion
Switched to a new branch 'colaboracion'

Albert@DESKTOP-F43V0VL MINGW64 /d/Varios/DS/fusion-remota (colaboracion)
$ echo "Colaboracion remota" > colaboracion.txt

Albert@DESKTOP-F43V0VL MINGW64 /d/Varios/DS/fusion-remota (colaboracion)
$ git add colaboracion.txt
warning: in the working copy of 'colaboracion.txt', LF will be replaced by CRL
the next time Git touches it

Albert@DESKTOP-F43V0VL MINGW64 /d/Varios/DS/fusion-remota (colaboracion)
$ git commit -m "add texto de colaboracion"
[colaboracion 2706277] add texto de colaboracion
 1 file changed, 1 insertion(+)
 create mode 100644 colaboracion.txt
```

3. Empuja los cambios a la rama remota:

```
Albert@DESKTOP-F43VOVL MINGW64 /d/Varios/DS/fusion-remota (colaboracion)
$ git push origin colaboracion
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 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
remote:
remote: Create a pull request for 'colaboracion' on GitHub by visiting:
remote:      https://github.com/Alkerg/fusion-remota/pull/new/colaboracion
remote:
To github.com:Alkerg/fusion-remota.git
 * [new branch]      colaboracion -> colaboracion
```

4. Simula una fusión desde la rama colaboracion en la rama main de otro colaborador. (Puedes usar la interfaz de GitHub para crear un Pull Request y realizar la fusión).



¿Cómo cambia la estrategia de fusión cuando colaboras con otras personas en un repositorio remoto?

Cuando colaboramos en un repositorio remoto debemos hacer pull request antes de hacer un merge, de tal forma que se eviten los conflictos antes de hacer la fusión

¿Qué problemas comunes pueden surgir al integrar ramas remotas?

Al integrar ramas remotas pueden ocurrir muchos errores de fusión al tener ramas desactualizadas respecto de las otras, en general problemas de sincronización.

Ejercicio final: flujo de trabajo completo

- Crea un proyecto con tres ramas: main, feature1, y feature2.
- Realiza varios cambios en feature1 y feature2 y simula colaboraciones paralelas.
- Realiza fusiones utilizando diferentes métodos:

- Fusiona feature1 con main utilizando git merge --ff.

```
Albert@DESKTOP-F43V0VL MINGW64 /d/Varios/DS/prueba-proyecto (main)
$ git merge --ff feature1
Updating bdf62d9..09f66e0
Fast-forward
 main.py | 3 ++-
 1 file changed, 2 insertions(+), 1 deletion(-)
```

- Fusiona feature2 con main utilizando git merge --no-ff.

```
Albert@DESKTOP-F43V0VL MINGW64 /d/Varios/DS/prueba-proyecto (main)
$ git merge --no-ff feature2
Auto-merging main.py
CONFLICT (content): Merge conflict in main.py
Automatic merge failed; fix conflicts and then commit the result.
```

- Haz una rama adicional llamada feature3 y aplasta sus commits utilizando git merge --squash.

```
Albert@DESKTOP-F43V0VL MINGW64 /d/Varios/DS/prueba-proyecto (main)
$ git merge --squash feature3
Updating 2c130bc..f79b0d2
Fast-forward
Squash commit -- not updating HEAD
 main.py | 5 +++++
 1 file changed, 5 insertions(+)
```

Analiza el historial de commits:

```
Albert@DESKTOP-F43V0VL MINGW64 /d/Varios/DS/prueba-proyecto (main)
$ git log --graph --oneline
* 82fe4fa (HEAD -> main) merge squash
*   2c130bc merge
|\
| * 0304ff9 (feature2) feature2: cambio 2
| * 761b39f feature2: cambio1
* | 09f66e0 (feature1) feature1 : cambio 1
|/
* bdf62d9 Commit inicial
```