

1. ¿Qué es GitHub?

Plataforma en línea para alojar repositorios Git. Permite colaboración, control de versiones y gestión de proyectos.

2. ¿Cómo crear un repositorio en GitHub?

Desde la terminal (usando GitHub CLI):

bash

Copy

gh repo create nombre-repo --public # Repositorio público

gh repo create nombre-repo --private # Repositorio privado

Si no tienes gh instalado:

bash

Copy

Crea un repositorio local y enlázalo después con GitHub

git init

git add .

git commit -m "Initial commit"

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

git push -u origin main

3. ¿Cómo crear una rama en Git?

bash

Copy

git branch nombre-rama # Crea la rama

git checkout nombre-rama # Cambia a la rama

O en un solo paso:

git checkout -b nombre-rama

4. ¿Cómo cambiar a una rama en Git?

bash

Copy

git checkout nombre-rama # Método tradicional

git switch nombre-rama # Alternativa moderna (Git 2.23+)

5. ¿Cómo fusionar ramas en Git?

bash

Copy

git checkout main # Cambia a la rama destino

git merge nombre-rama # Fusiona la rama

6. ¿Cómo crear un commit en Git?

bash

Copy

```
git add .          # Añade todos los cambios al staging
git commit -m "Mensaje"
```

7. ¿Cómo enviar un commit a GitHub?

bash

Copy

```
git push origin nombre-rama # Sube cambios a la rama remota
```

8. ¿Qué es un repositorio remoto?

Versión del repositorio alojada en un servidor (como GitHub). Se gestiona con git remote.
(No requiere comando).

9. ¿Cómo agregar un repositorio remoto a Git?

bash

Copy

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

10. ¿Cómo empujar cambios a un repositorio remoto?

bash

Copy

```
git push -u origin main # -u establece la rama por defecto
```

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

bash

Copy

```
git pull origin main # Descarga cambios y fusiona
```

12. ¿Qué es un fork de repositorio?

Copia personal de un repositorio ajeno en tu cuenta de GitHub.
(No requiere comando).

13. ¿Cómo crear un fork de un repositorio?

No se puede hacer desde la terminal (requiere la interfaz web de GitHub).

Para clonar tu fork:

bash

Copy

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

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

Desde la terminal (usando GitHub CLI):

bash

Copy

```
gh pr create --base main --head tu-rama --title "Título" --body "Descripción"
```

15. ¿Cómo aceptar una solicitud de extracción?

No se puede desde la terminal (se hace desde la interfaz web de GitHub).

16. ¿Qué es una etiqueta (tag) en Git?

Marca un punto específico en el historial (ej: versiones).

(No requiere comando).

17. ¿Cómo crear una etiqueta en Git?

bash

Copy

```
git tag -a v1.0.0 -m "Versión 1.0" # Etiqueta anotada
```

18. ¿Cómo enviar una etiqueta a GitHub?

bash

Copy

```
git push --tags # Envía todas las etiquetas
```

```
git push origin v1.0.0 # Envía una etiqueta específica
```

19. ¿Qué es un historial de Git?

Registro de todos los commits realizados.

(No requiere comando).

20. ¿Cómo ver el historial de Git?

bash

Copy

```
git log          # Muestra el historial completo
```

```
git log --oneline # Versión resumida
```

```
git log --graph   # Muestra ramas y fusiones
```

21. ¿Cómo buscar en el historial de Git?

bash

Copy

```
git log --grep "palabra-clave" # Busca en mensajes de commit
```

22. ¿Cómo borrar el historial de Git?

No recomendado (destructivo). Para borrar commits locales:

bash

Copy

```
git reset --hard HEAD~3 # Elimina los últimos 3 commits
```

23. ¿Qué es un repositorio privado en GitHub?

Solo accesible para colaboradores invitados.

(No requiere comando).

24. ¿Cómo crear un repositorio privado en GitHub?

Con GitHub CLI:

bash

Copy

```
gh repo create nombre-repo --private
```

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

No se puede desde la terminal (se hace desde la web de GitHub: Settings > Collaborators).

26. ¿Qué es un repositorio público en GitHub?

Visible para todos.

(No requiere comando).

27. ¿Cómo crear un repositorio público en GitHub?

Con GitHub CLI:

bash

Copy

```
gh repo create nombre-repo --public
```

28. ¿Cómo compartir un repositorio público en GitHub?

Proporciona el enlace HTTPS/SSH del repositorio. Para clonar:

bash

Copy

```
git clone https://github.com/usuario/repo-publico.git
```

EJERCICIO 2

```
Yamatino@DESKTOP-PND3GLC MINGW64 /z/PROGRAMAR/UTN-TUPaD-P1/02 - Trabajo Colabora
tivo (main)
$ git add .

Yamatino@DESKTOP-PND3GLC MINGW64 /z/PROGRAMAR/UTN-TUPaD-P1/02 - Trabajo Colabora
tivo (main)
$ git commit -m "agregando mi-archivo.txt"
[main c3b53e2] agregando mi-archivo.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 02 - Trabajo Colaborativo/mi-archivo.txt

Yamatino@DESKTOP-PND3GLC MINGW64 /z/PROGRAMAR/UTN-TUPaD-P1/02 - Trabajo Colabora
tivo (main)
$ git push origin main
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 450 bytes | 450.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Yamatino/UTN-TUPaD-P1.git
61ce812..c3b53e2 main -> main

Yamatino@DESKTOP-PND3GLC MINGW64 /z/PROGRAMAR/UTN-TUPaD-P1/02 - Trabajo Colabora
tivo (main)
$ |
```

```
MINGW64:/z/PROGRAMAR/UTN-TUPaD-P1

Yamatino@DESKTOP-PND3GLC MINGW64 /z/PROGRAMAR/UTN-TUPaD-P1 (main)
$ git checkout -b nuevaBranch
Switched to a new branch 'nuevaBranch'

Yamatino@DESKTOP-PND3GLC MINGW64 /z/PROGRAMAR/UTN-TUPaD-P1 (nuevaBranch)
$ git branch
  main
* nuevaBranch

Yamatino@DESKTOP-PND3GLC MINGW64 /z/PROGRAMAR/UTN-TUPaD-P1 (nuevaBranch)
$ echo "modificacion a archivo" > mi.archivo.txt

Yamatino@DESKTOP-PND3GLC MINGW64 /z/PROGRAMAR/UTN-TUPaD-P1 (nuevaBranch)
$ git add .
warning: in the working copy of 'mi.archivo.txt', LF will be replaced by CRLF the next time Git touches it

Yamatino@DESKTOP-PND3GLC MINGW64 /z/PROGRAMAR/UTN-TUPaD-P1 (nuevaBranch)
$ git status
On branch nuevaBranch
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   mi.archivo.txt

Yamatino@DESKTOP-PND3GLC MINGW64 /z/PROGRAMAR/UTN-TUPaD-P1 (nuevaBranch)
$ git commit -m "Agrego archivo desde nuevaBranch"
[nuevaBranch f111292] Agrego archivo desde nuevaBranch
 1 file changed, 1 insertion(+)
 create mode 100644 mi.archivo.txt

Yamatino@DESKTOP-PND3GLC MINGW64 /z/PROGRAMAR/UTN-TUPaD-P1 (nuevaBranch)
$ git push origin nuevaBranch
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 313 bytes | 313.00 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 'nuevaBranch' on GitHub by visiting:
remote:   https://github.com/Yamatino/UTN-TUPaD-P1/pull/new/nuevaBranch
remote:
To https://github.com/Yamatino/UTN-TUPaD-P1.git
 * [new branch]   nuevaBranch -> nuevaBranch

Yamatino@DESKTOP-PND3GLC MINGW64 /z/PROGRAMAR/UTN-TUPaD-P1 (nuevaBranch)
$ |
```

Branches

Overview

Yours

Active

Stale

All

Search branches...

Default

Branch

main



Your branches

Branch

nuevaBranch



Active branches

Branch

nuevaBranch



EJERCICIO 3 (<https://github.com/Yamatino/conflict-exercise>)

```
MINGW64:/z/PROGRAMAR/conflict-exercise

Yamatino@DESKTOP-PND3GLC MINGW64 /z/PROGRAMAR
$ git clone https://github.com/Yamatino/conflict-exercise~
Cloning into 'conflict-exercise~'...
remote: Repository not found.
fatal: repository 'https://github.com/Yamatino/conflict-exercise~/' not found

Yamatino@DESKTOP-PND3GLC MINGW64 /z/PROGRAMAR
$ git clone https://github.com/Yamatino/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.

Yamatino@DESKTOP-PND3GLC MINGW64 /z/PROGRAMAR
$ cd conflict-exercise/

Yamatino@DESKTOP-PND3GLC MINGW64 /z/PROGRAMAR/conflict-exercise (main)
$ git checkout -b feature-branch
Switched to a new branch 'feature-branch'

Yamatino@DESKTOP-PND3GLC MINGW64 /z/PROGRAMAR/conflict-exercise (feature-branch)
$ git add readme.md

Yamatino@DESKTOP-PND3GLC MINGW64 /z/PROGRAMAR/conflict-exercise (feature-branch)
$ git status
On branch feature-branch
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")

Yamatino@DESKTOP-PND3GLC MINGW64 /z/PROGRAMAR/conflict-exercise (feature-branch)
$ git commit -m "added a line in feature-branch"
On branch feature-branch
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")

Yamatino@DESKTOP-PND3GLC MINGW64 /z/PROGRAMAR/conflict-exercise (feature-branch)
$ git add README.md

Yamatino@DESKTOP-PND3GLC MINGW64 /z/PROGRAMAR/conflict-exercise (feature-branch)
$ git status
On branch feature-branch
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README.md

Yamatino@DESKTOP-PND3GLC MINGW64 /z/PROGRAMAR/conflict-exercise (feature-branch)
$ git commit -m "added a line in feature-branch"
[feature-branch 17ee5ab] added a line in feature-branch
1 file changed, 2 insertions(+)

Yamatino@DESKTOP-PND3GLC MINGW64 /z/PROGRAMAR/conflict-exercise (feature-branch)
$
```

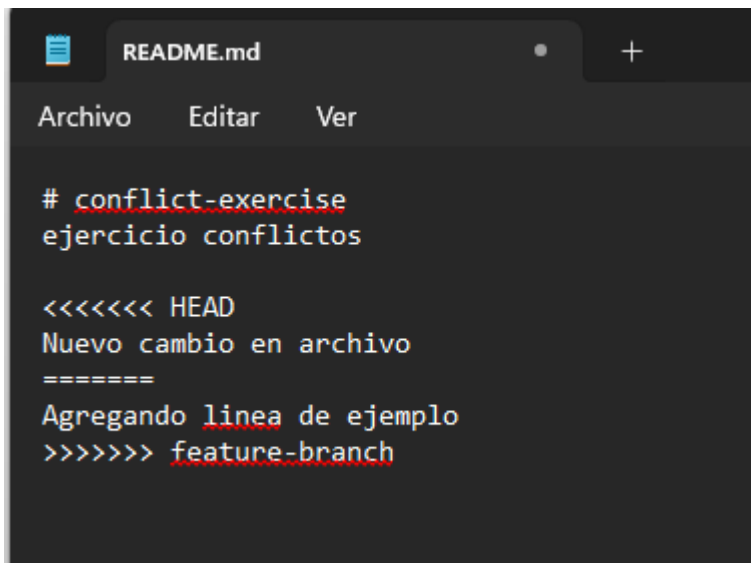


```
Yamatino@DESKTOP-PND3GLC MINGW64 /z/PROGRAMAR/conflict-exercise (feature-branch)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

Yamatino@DESKTOP-PND3GLC MINGW64 /z/PROGRAMAR/conflict-exercise (main)
$ git add README.md

Yamatino@DESKTOP-PND3GLC MINGW64 /z/PROGRAMAR/conflict-exercise (main)
$ git commit -m "added a line in main branch"
[main 847879a] added a line in main branch
1 file changed, 2 insertions(+)

Yamatino@DESKTOP-PND3GLC MINGW64 /z/PROGRAMAR/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.
```



```
README.md

Archivo  Editar  Ver

# conflict-exercise
ejercicio conflictos

<<<<<<< HEAD
Nuevo cambio en archivo
=====
Agregando linea de ejemplo
>>>>>>> feature-branch
```

```
Yamatino@DESKTOP-PND3GLC MINGW64 /z/PROGRAMAR/conflict-exercise (main|MERGING)
$ git add README.md

Yamatino@DESKTOP-PND3GLC MINGW64 /z/PROGRAMAR/conflict-exercise (main|MERGING)
$ git commit -m "resolved merge conflict"
[main 8007430] resolved merge conflict

Yamatino@DESKTOP-PND3GLC MINGW64 /z/PROGRAMAR/conflict-exercise (main)
$ git push origin main
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 16 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (9/9), 879 bytes | 879.00 KiB/s, done.
Total 9 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Yamatino/conflict-exercise.git
   ba4b704..8007430  main -> main

Yamatino@DESKTOP-PND3GLC MINGW64 /z/PROGRAMAR/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/Yamatino/conflict-exercise/pull/new/feature-branch
remote:
To https://github.com/Yamatino/conflict-exercise.git
 * [new branch]      feature-branch -> feature-branch

Yamatino@DESKTOP-PND3GLC MINGW64 /z/PROGRAMAR/conflict-exercise (main)
$ |
```